Projektplanung mit partiell erneuerbaren Ressourcen

Entwicklung und Untersuchung von Branch-and-Bound-Verfahren

Dissertation

zur Erlangung des Doktorgrades
der Wirtschaftswissenschaften

vorgelegt von

Kai Watermeyer, M. Sc.
aus Hildesheim

genehmigt von der
Fakultät für Energie und Wirtschaftswissenschaften
der Technischen Universität Clausthal

Tag der mündlichen Prüfung:

17. Dezember 2020

Dissertation, Technische Universität Clausthal, 2020

D 104

Dekan: Prof. Dr. rer. nat. Bernd Lehmann

Vorsitzender der Promotionskommission: Prof. Dr. sc. pol. Roland Menges

Betreuer und Erstgutachter: Prof. Dr. rer. pol. Jürgen Zimmermann

Weiterer Gutachter: Prof. Dr. rer. pol. Christoph Schwindt

Vorwort

Die vorliegende Dissertation ist während meiner Tätigkeit als wissenschaftlicher Mitarbeiter an der Abteilung für Betriebswirtschaftslehre und Unternehmensforschung am Institut für Wirtschaftswissenschaft der Technischen Universität Clausthal entstanden.

Ein besonderer Dank gilt meinem Doktorvater Prof. Dr. Jürgen Zimmermann, der mir nicht nur die Möglichkeit zur Promotion eröffnet hat, sondern darüber hinaus auch stets zur fachlichen Diskussion bereit stand. Herrn Prof. Dr. Christoph Schwindt danke ich für die Übernahme des Korreferats sowie für seine interessanten Anregungen bezüglich weiterführender Forschungsmöglichkeiten.

Weiterhin möchte ich mich ganz besonders bei all denjenigen bedanken, die mich auf meinem Weg der Promotion begleitet und unterstützt haben. Gemeinsame Mittags- und Kaffeepausen sowie gemeinsame Unternehmungen haben die Zeit am Institut in besonderer Weise bereichert. Meinen heutigen und ehemaligen Kollegen danke ich darüber hinaus für die konstruktive und freundschaftliche Zusammenarbeit. Für die kritische Durchsicht des Manuskripts danke ich Herrn Philipp Aschersleben, Frau Dr. Anja Heßler, Frau Mareike Karnebogen und Herrn Max Reinke.

Abschließend danke ich meiner Familie, die mich zu jeder Zeit unterstützt hat.

Goslar, Februar 2021 *Kai Watermeyer*

Zusammenfassung

Die Implementierung eines geeigneten und zielgerichteten Projektmanagements stellt für viele Unternehmen im Hinblick auf kürzer werdende Innovationszyklen und sich verändernde Marktanforderungen einen immer wichtigeren Erfolgsfaktor dar. Eine entscheidende Bedeutung kommt dabei vor allem der Projektplanung als Bindeglied zwischen der Vorbereitungs- und der Ausführungsphase eines Projekts zu. Insbesondere die ressourcenbeschränkte Projektplanung kann durch die Bestimmung effizienter und kostengünstiger Einsatzpläne für begrenzt verfügbare Ressourcen einen wichtigen Beitrag zur Wettbewerbsfähigkeit eines Unternehmens leisten.

Die meisten Modelle der ressourcenbeschränkten Projektplanung gehen vereinfachend davon aus, dass erneuerbare Ressourcen in bestimmten Mengen in jeder Zeitperiode für die Ausführung von Vorgängen zur Verfügung stehen, die durch Vorrangbeziehungen miteinander verbunden sind. Diese einschränkenden Annahmen führen jedoch dazu, dass praxisrelevante Restriktionen wie Arbeitszeitvereinbarungen oder Vorgaben zur Höchstauslastung von Maschinen durch Modelle der ressourcenbeschränkten Projektplanung nicht abgebildet werden können. Eine Möglichkeit, um komplexere Restriktionen in die Modelle einzubinden, stellen sogenannte partiell erneuerbare Ressourcen dar, die Kapazitätsrestriktionen auch über mehrere Zeitperioden modellieren können. Durch diese Art von Ressourcen können unter anderem maximale Arbeitsstunden am Wochenende oder vorgeschriebene Pausenzeiten von Arbeitskräften modelliert werden, die durch klassische Modelle der Projektplanung nicht abgebildet werden können. Weitere praxisrelevante Restriktionen wie technologisch bedingte Zeitfenster für die Ausführung von Fertigungsprozessen können zudem durch zeitliche Mindest- und Höchstabstände bzw. durch allgemeine Zeitbeziehungen zwischen den Vorgängen eines Projekts dargestellt werden.

In der vorliegenden Arbeit wird das Projektdauerminimierungsproblem mit allgemeinen Zeitbeziehungen und partiell erneuerbaren Ressourcen (RCPSP/max-π) untersucht. Ein Schwerpunkt liegt dabei auf der Entwicklung von Branch-and-Bound-Verfahren, die auf unterschiedlichen Enumerationsschemata basieren. Es werden zwei relaxationsbasierte und ein konstruktionsbasiertes Branch-and-Bound-Verfahren vorgestellt, deren Leistungsfähigkeit anhand geeigneter Testinstanzen durch eine experimentelle Performance-Analyse untersucht werden. Die Ergebnisse der Analysen zeigen, dass eines der relaxati-

onsbasierten Verfahren, das die zeitzulässigen Startzeitpunkte der Vorgänge des Projekts schrittweise in disjunkte Mengen zerlegt, die beiden anderen Ansätze dominiert. Aus einem weiterführenden Vergleich mit dem MILP-Solver IBM CPLEX sowie den besten bislang bekannten Näherungsverfahren zur Projektdauerminimierung mit partiell erneuerbaren Ressourcen wird zudem die vorteilhafte Performance des dominanten Branch-and-Bound-Verfahrens bestätigt.

In der vorliegenden Arbeit wird weiterhin gezeigt, dass der Einsatz partiell erneuerbarer Ressourcen ein weites Feld an Modellierungsmöglichkeiten eröffnet, das auch andere Konzepte der Projektplanung umfasst, die über die letzten Jahrzehnte entwickelt wurden. Basierend auf diesen Ergebnissen wird zudem gezeigt, dass exakte Verfahren für das RCPSP/max-π auch zur Lösung anderer bekannter Projektdauerminimierungsprobleme aus der Literatur eingesetzt werden können.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ACO	Absolute Capacity Overrun
AMAR	Average Maximal Additional Resource Consumption
CBB	Konstruktionsbasiertes Branch-and-Bound-Verfahren
DFS	Depth-First Search
DST	Delayed Start Time
EFF	Early Free Float
ILP	Ganzzahliges lineares Programm (engl. Integer Linear Program)
LIFO	Last-In-First-Out
LST	Latest Start Time
LT	Lowest Time
MILP	Gemischt-ganzzahliges lineares Programm (engl. Mixed-Integer Linear Program)
MRC	Maximal Resource Consumption
NCA	Number of Consuming Activities
PBB	Partitionsbasiertes Branch-and-Bound-Verfahren
PF	Path Following
PV	Priority Value
RBB	Relaxationsbasiertes Branch-and-Bound-Verfahren
RCO	Relative Capacity Overrun
RCPSP	Ressourcenbeschränktes Projektplanungsproblem (engl. Resource-Constrained Project Scheduling Problem)
RCPSP/max	RCPSP mit allgemeinen Zeitbeziehungen
RCPSP/max-c	RCPSP/max mit diskreten kumulativen Ressourcen
RCPSP/max-cal	RCPSP/max mit Kalendern
RCPSP/max-cc	RCPSP/max mit kontinuierlichen kumulativen Ressourcen
RCPSP/max-π	RCPSP/max mit partiell erneuerbaren Ressourcen
RCPSP/π	RCPSP mit partiell erneuerbaren Ressourcen
RCR	Remaining Capacity Reduction

RU	Resource Usage
RUT	Resource Usage per Time
SPS	Scattered-Path-Suche
ST	Slack Time
TF	Total Float
TMAR	Total Maximal Additional Resource Consumption
TS	Total Successor
ULT	Usage-Limitation-Technik
UPT	Usage-Preserving-Technik

Symbolverzeichnis

$\emptyset$	Leere Menge
0	Vorgang „Projektbeginn“
$\mathcal{AS}$	Menge der aktiven Schedules
$\mathcal{B}$	Anzahl der Startzeitunterbrechungen der Startzeitbeschränkung W
$\mathcal{B}_i$	Anzahl der Startzeitunterbrechungen der Startzeitbeschränkung W_i von Vorgang $i \in V$
$\mathcal{C}$	Menge der aktuell eingeplanten Vorgänge eines Projekts
$\bar{\mathcal{C}}$	Menge der aktuell nicht eingeplanten Vorgänge eines Projekts
$\bar{d}$	Vorgegebene maximale Projektdauer
d_{ij}	Länge eines längsten Wegs von Vorgang $i \in V$ zu Vorgang $j \in V$ in Projektnetzplan N
$\widetilde{d}_{ij}(W,t)$	Zeitabstand zwischen Zeitpunkt t und dem frühestmöglichen W-zulässigen Startzeitpunkt von Vorgang $j \in V$, falls Vorgang $i \in V$ nicht vor Zeitpunkt t startet; $\widetilde{d}_{ij}(W,t) := ES_j(W,i,t) - t$
$\widehat{d}_{ij}(W,t)$	Zeitabstand zwischen Zeitpunkt t und dem spätestmöglichen W-zulässigen Startzeitpunkt von Vorgang $j \in V$, falls Vorgang $i \in V$ nicht nach Zeitpunkt t startet; $\widehat{d}_{ij}(W,t) := LS_j(W,i,t) - t$
δ	Vektor der Zeitabstände $\delta := (\delta_{ij})_{(i,j)\in E}$
δ_{ij}	Zeitabstand zwischen den Startzeitpunkten der Vorgänge $i,j \in V$ für Vorgangspaar $(i,j) \in E$
$\widetilde{\delta}_{ij}(W,t)$	Zeitabstand zwischen dem frühesten Startzeitpunkt $\tau \in W_j$ von Vorgang $j \in V$, der die Bedingung $\tau \geq t + \delta_{ij}$ erfüllt, und Startzeitpunkt t von Vorgang $i \in V$ für Vorgangspaar $(i,j) \in E$
$\widehat{\delta}_{ij}(W,t)$	Zeitabstand zwischen dem spätesten Startzeitpunkt $\tau \in W_i$ von Vorgang $i \in V$, der die Bedingung $\tau \leq t - \delta_{ij}$ erfüllt, und Startzeitpunkt t von Vorgang $j \in V$ für Vorgangspaar $(i,j) \in E$
D	Distanzmatrix $D := (d_{ij})_{i,j\in V}$
$\widetilde{D}(W)$	Mindestabstandsmatrix $\widetilde{D}(W) := ([\widetilde{d}_{ij}(W,t)])_{i,j\in V}$
$\widehat{D}(W)$	Höchstabstandsmatrix $\widehat{D}(W) := ([\widehat{d}_{ij}(W,t)])_{i,j\in V}$
$\Delta^u_{ik}(t)$	Veränderung der Belegung von Ressource $k \in \mathcal{R}$ durch Vorgang $i \in V$, falls der Startzeitpunkt t von Vorgang i um eine Zeiteinheit erhöht wird; $\Delta^u_{ik}(t) := r^u_{ik}(t+1) - r^u_{ik}(t)$

$e_i(t)$	Endzeitpunkt des Startzeitintervalls I der Startzeitbeschränkung W_i von Vorgang $i \in V$, dem Zeitpunkt t zugeordnet ist ($t \in I$)
E	Menge der Vorgangspaare eines Projekts, zwischen denen eine Zeitbeziehung vorliegt; $E \subset V \times V$
ES	Frühester zeitzulässiger Schedule
ES_i	Frühester zeitzulässiger Startzeitpunkt von Vorgang $i \in V$
$ES(W)$	Frühester W-zulässiger Schedule
$ES_i(W)$	Frühester W-zulässiger Startzeitpunkt von Vorgang $i \in V$
$ES(W, \alpha, t_\alpha)$	Frühestmöglicher W-zulässiger Schedule, falls Vorgang $\alpha \in V$ nicht vor Zeitpunkt t_α startet
$ES_i(W, \alpha, t_\alpha)$	Frühestmöglicher W-zulässiger Startzeitpunkt von Vorgang $i \in V$, falls Vorgang $\alpha \in V$ nicht vor Zeitpunkt t_α startet
$\mathcal{E}$	Menge der einplanbaren Vorgänge in einer Iteration eines Konstruktionsverfahrens; $\mathcal{E} \subseteq \bar{\mathcal{C}}$
γ	Domain-Konsistenztest, $(\gamma(W))_i \subseteq W_i$ für alle Vorgänge $i \in V$
Γ	Menge aus Domain-Konsistenztests
$\mathcal{H}$	Menge aller ganzzahligen Zeitpunkte des Planungshorizonts; $\mathcal{H} := \{0, 1, \ldots, \bar{d}\}$
$\mathcal{I}$	Anzahl der Belegungsintervalle in den Mengen Π_k über alle Ressourcen $k \in \mathcal{R}$
$\mathcal{I}_k$	Anzahl der Belegungsintervalle in der Menge Π_k der Ressource $k \in \mathcal{R}$
LB	Untere Schranke für die kürzeste Projektdauer über alle zulässigen Schedules im Suchraum eines Enumerationsknotens
$LB0^\pi$	Untere Schranke für die kürzeste Projektdauer über alle Schedules $S \in \mathcal{S}(W)$; $LB0^\pi := ES_{n+1}(W)$
LBD^π	Destruktive untere Schranke für die kürzeste Projektdauer über alle Schedules $S \in \mathcal{S}(W)$
LS	Spätester zeitzulässiger Schedule
LS_i	Spätester zeitzulässiger Startzeitpunkt von Vorgang $i \in V$
$LS(W)$	Spätester W-zulässiger Schedule
$LS_i(W)$	Spätester W-zulässiger Startzeitpunkt von Vorgang $i \in V$
$LS(W, \alpha, t_\alpha)$	Spätestmöglicher W-zulässiger Schedule, falls Vorgang $\alpha \in V$ nicht nach Zeitpunkt t_α startet
$LS_i(W, \alpha, t_\alpha)$	Spätestmöglicher W-zulässiger Startzeitpunkt von Vorgang $i \in V$, falls Vorgang $\alpha \in V$ nicht nach Zeitpunkt t_α startet
Λ	Liste der generierten direkten Nachfolger eines Enumerationsknotens in einem Branch-and-Bound-Verfahren
$n + 1$	Vorgang „Projektende“
N	Projektnetzplan $N := (V, E, \delta)$

ν	Markierung aller Vorgänge $i \in V$; $\nu := (\nu_i)_{i \in V}$
ν_i	Markierung des Vorgangs $i \in V$
$\mathcal{O}$	Landausches Symbol; für $f, g : \mathbb{N}^k \mapsto \mathbb{R}_{\geq 0}$ gelte, dass $g \in \mathcal{O}(f)$, wenn eine Konstante $c > 0$ und eine positive ganze Zahl n existieren, sodass $g(x) \leq cf(x)$ für alle x mit $x_i \geq n$ für ein $i = 1, 2, \ldots, k$
$\mathcal{OS}$	Menge der optimalen Schedules
Ω	Menge der noch zu untersuchenden Knoten innerhalb eines Enumerationsverfahrens
$\bar{p}$	Längste Ausführungsdauer über alle Vorgänge $i \in V$; $\bar{p} := \max_{i \in V} p_i$
p_i	Ausführungsdauer von Vorgang $i \in V$
$Pred(i)$	Menge der direkten Vorgänger von Vorgang $i \in V$ in Projektnetzplan N; $Pred(i) := \{h \in V \mid (h, i) \in E\}$
$\mathcal{P}$	Menge aller Zeitperioden des Planungshorizonts; $\mathcal{P} := \{1, 2, \ldots, \bar{d}\}$
Φ	Menge der generierten Schedules innerhalb eines Enumerationsverfahrens
Π_k	Menge aller Perioden, in denen Ressource $k \in \mathcal{R}$ durch die Ausführung eines Vorgangs in Anspruch genommen wird
$r_k^c(S)$	Inanspruchnahme der Ressource $k \in \mathcal{R}$ gemäß Schedule S; $r_k^c(S) := \sum_{i \in V_k} r_{ik}^c(S_i)$
$r_k^c(S^{\mathcal{C}})$	Inanspruchnahme der Ressource $k \in \mathcal{R}$ gemäß Teilschedule $S^{\mathcal{C}}$; $r_k^c(S^{\mathcal{C}}) := \sum_{i \in \mathcal{C}} r_{ik}^c(S_i)$
$\underline{r}_k^c(W)$	Summe der Mindestinanspruchnahmen $\underline{r}_{ik}^c(W)$ der Ressource $k \in \mathcal{R}$ durch alle Vorgänge $i \in V_k$
$r_{ik}^c(S_i)$	Inanspruchnahme der Ressource $k \in \mathcal{R}$ durch Vorgang $i \in V$ in Abhängigkeit vom Startzeitpunkt S_i; $r_{ik}^c(S_i) := r_{ik}^u(S_i)\, r_{ik}^d$
$\underline{r}_{ik}^c(W)$	Mindestinanspruchnahme der Ressource $k \in \mathcal{R}$ durch Vorgang $i \in V_k$ über alle Startzeitpunkte $t \in W_i \cap [ES_i(W), LS_i(W)]$
$r_{ik}^{c,min}(W)$	Mindestinanspruchnahme der Ressource $k \in \mathcal{R}$ durch Vorgang $i \in V$ über alle Startzeitpunkte $t \in W_i$
$r_{ikt}^{c,min}(W, D)$	Summe der Mindestinanspruchnahmen $r_{ijkt}^{c,min}(W, D)$ der Ressource $k \in \mathcal{R}$ durch alle Vorgänge $j \in V \setminus \{i\}$ und der Ressourceninanspruchnahme $r_{ik}^c(t)$ für den Startzeitpunkt $t \in W_i$ von Vorgang $i \in V$
$r_{ijkt}^{c,min}(W, D)$	Mindestinanspruchnahme der Ressource $k \in \mathcal{R}$ durch Vorgang $j \in V \setminus \{i\}$ über alle Startzeitpunkte $\tau \in W_j \cap [t + d_{ij}, t - d_{ji}]$ für einen Startzeitpunkt $t \in W_i$ von Vorgang $i \in V$
$r_k^{c,min}(W, \underline{S}, \overline{S})$	Summe der Mindestinanspruchnahmen $r_{ik}^{c,min}(W, \underline{S}_i, \overline{S}_i)$ der Ressource $k \in \mathcal{R}$ durch alle Vorgänge $i \in V_k$ für die Schedules $\underline{S}$ und $\overline{S}$
$r_{ik}^{c,min}(W, \underline{S}_i, \overline{S}_i)$	Mindestinanspruchnahme der Ressource $k \in \mathcal{R}$ durch Vorgang $i \in V$ über alle Startzeitpunkte $t \in W_i \cap [\underline{S}_i, \overline{S}_i]$

$r_{ikt}^{c,min}(W, \widetilde{D}, \widehat{D})$	Summe der Mindestinanspruchnahmen $r_{ijkt}^{c,min}(W, \widetilde{D}, \widehat{D})$ der Ressource $k \in \mathcal{R}$ durch alle Vorgänge $j \in V \setminus \{i\}$ und der Ressourceninanspruchnahme $r_{ik}^{c}(t)$ für den Startzeitpunkt $t \in W_i$ von Vorgang $i \in V$
$r_{ijkt}^{c,min}(W, \widetilde{D}, \widehat{D})$	Mindestinanspruchnahme der Ressource $k \in \mathcal{R}$ durch Vorgang $j \in V \setminus \{i\}$ über alle Startzeitpunkte $\tau \in W_j \cap [t+\widetilde{d}_{ij}(W,t), t+\widehat{d}_{ij}(W,t)]$ für einen Startzeitpunkt $t \in W_i$ von Vorgang $i \in V$
r_{ik}^{d}	Bedarf an Ressource $k \in \mathcal{R}$ durch Vorgang $i \in V$ in jeder Periode in Π_k, in der Vorgang i ausgeführt wird
$r_{ik}^{u}(S_i)$	Anzahl der Perioden in Π_k, in denen sich Vorgang $i \in V$ in Abhängigkeit vom Startzeitpunkt S_i in Ausführung befindet (Ressourcenbelegung)
$\underline{r}_{ik}^{u}(W)$	Mindestbelegung der Ressource $k \in \mathcal{R}$ durch Vorgang $i \in V_k$ über alle Startzeitpunkte $t \in W_i \cap [ES_i(W), LS_i(W)]$
$r_{ik}^{u,min}(W)$	Mindestbelegung der Ressource $k \in \mathcal{R}$ durch Vorgang $i \in V$ über alle Startzeitpunkte $t \in W_i$
$r_{ijkt}^{u,min}(W, D)$	Mindestbelegung der Ressource $k \in \mathcal{R}$ durch Vorgang $j \in V \setminus \{i\}$ über alle Startzeitpunkte $\tau \in W_j \cap [t + d_{ij}, t - d_{ji}]$ für einen Startzeitpunkt $t \in W_i$ von Vorgang $i \in V$
$r_{ijkt}^{u,min}(W, \widetilde{D}, \widehat{D})$	Mindestbelegung der Ressource $k \in \mathcal{R}$ durch Vorgang $j \in V \setminus \{i\}$ über alle Startzeitpunkte $\tau \in W_j \cap [t + \widetilde{d}_{ij}(W,t), t + \widehat{d}_{ij}(W,t)]$ für einen Startzeitpunkt $t \in W_i$ von Vorgang $i \in V$
R_k	Kapazität der partiell erneuerbaren Ressource $k \in \mathcal{R}$
$\mathcal{R}$	Menge der partiell erneuerbaren Ressourcen eines Projekts
$\mathcal{R}_i$	Menge der partiell erneuerbaren Ressourcen mit $r_{ik}^{d} > 0$ für den Vorgang $i \in V$; $\mathcal{R}_i := \{k \in \mathcal{R} \mid r_{ik}^{d} > 0\}$
$\mathcal{R}^c(S)$	Menge der partiell erneuerbaren Ressourcen mit $r_k^c(S) > R_k$ für einen gegebenen Schedule S; $\mathcal{R}^c(S) := \{k \in \mathcal{R} \mid r_k^c(S) > R_k\}$
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}_{\geq 0}$	Menge der nicht-negativen reellen Zahlen
$s_i(t)$	Startzeitpunkt des Startzeitintervalls I der Startzeitbeschränkung W_i von Vorgang $i \in V$, dem Zeitpunkt t zugeordnet ist ($t \in I$)
sgn	Signumfunktion; $\mathrm{sgn}(x) := \lvert x \rvert / x$, falls $x \neq 0$; $\mathrm{sgn}(x) := x$, sonst
S	Vektor der Startzeitpunkte S_i aller Vorgänge $i \in V$ (Schedule)
$S^{\mathcal{C}}$	Teilschedule $S^{\mathcal{C}} := (S_i)_{i \in \mathcal{C}}$
S_i	Startzeitpunkt von Vorgang $i \in V$
$Succ(i)$	Menge der direkten Nachfolger von Vorgang $i \in V$ in Projektnetzplan N; $Succ(i) := \{j \in V \mid (i,j) \in E\}$
$\mathcal{S}$	Menge der zulässigen Schedules
$\mathcal{S}_R$	Menge der ressourcenzulässigen Schedules

$\mathcal{S}_T$	Menge der zeitzulässigen Schedules
$\mathcal{S}(W)$	Menge der zulässigen Schedules, die W-zulässig sind; $\mathcal{S}(W) := \mathcal{S}_T(W) \cap \mathcal{S}$
$\mathcal{S}_T(W)$	Menge der W-zulässigen Schedules; $\mathcal{S}_T(W) := \{S \in \mathcal{S}_T \mid S_i \in W_i \text{ für alle } i \in V\}$
$\mathcal{S}_T^{UB}(W)$	Menge der W-zulässigen Schedules mit $S_{n+1} < UB$; $\mathcal{S}_T^{UB}(W) := \widehat{\mathcal{S}}_T(W, n+1, UB-1)$
$\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha)$	Menge der W-zulässigen Schedules mit $S_\alpha \geq t_\alpha$; $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha) := \{S \in \mathcal{S}_T(W) \mid S_\alpha \geq t_\alpha\}$
$\widehat{\mathcal{S}}_T(W, \alpha, t_\alpha)$	Menge der W-zulässigen Schedules mit $S_\alpha \leq t_\alpha$; $\widehat{\mathcal{S}}_T(W, \alpha, t_\alpha) := \{S \in \mathcal{S}_T(W) \mid S_\alpha \leq t_\alpha\}$
T_i	Eingeschränkte Menge der Einplanungszeitpunkte von Vorgang $i \in V$
$\mathcal{T}_i$	Menge aller zeitzulässigen Startzeitpunkte von Vorgang $i \in V$; $\mathcal{T}_i := [ES_i, LS_i] \cap \mathbb{Z}$
Θ_i	Menge der Einplanungszeitpunkte von Vorgang $i \in V$
$\bar{u}_{ik}$	Obere Schranke für die Belegung von Ressource $k \in \mathcal{R}$ durch Vorgang $i \in V$; $r_{ik}^u(S_i) \leq \bar{u}_{ik}$
UB	Obere Schranke für die kürzeste Projektdauer über alle zulässigen Schedules $S \in \mathcal{S}$
V	Menge der Vorgänge eines Projekts
V^e	Menge der fiktiven Vorgänge (Ereignisse) eines Projekts
V^r	Menge der realen Vorgänge eines Projekts
V_k	Menge der Vorgänge eines Projekts mit einem Bedarf nach Ressource $k \in \mathcal{R}$; $V_k := \{i \in V \mid r_{ik}^d > 0\}$
$V_k(S)$	Menge der Vorgänge, die Ressource $k \in \mathcal{R}$ für einen gegebenen Schedule S in Anspruch nehmen; $V_k(S) := \{i \in V_k \mid r_{ik}^u(S_i) > 0\}$
W	Startzeitbeschränkung $W := (W_i)_{i \in V}$
W_i	Startzeitbeschränkung von Vorgang $i \in V$; $W_i \subseteq \mathcal{H}$
$W_{ik}(\bar{u}_{ik})$	Menge aller zeitzulässigen Startzeitpunkte $t \in \mathcal{T}_i$ von Vorgang $i \in V$ mit einer Belegung von Ressource $k \in \mathcal{R}$, die $\bar{u}_{ik}$ nicht übersteigt; $W_{ik}(\bar{u}_{ik}) := \{t \in \mathcal{T}_i \mid r_{ik}^u(t) \leq \bar{u}_{ik}\}$
$W_{ij}^e(t)$	Menge aller Startzeitpunkte $\tau \in W_j$ von Vorgang $j \in V$, für die $t + d_{ij} \leq \tau \leq t - d_{ji}$ gilt
$\mathbb{Z}$	Menge der ganzen Zahlen
$\mathbb{Z}_{\geq 0}$	Menge der nicht-negativen ganzen Zahlen
$\mathbb{Z}_{>0}$	Menge der positiven ganzen Zahlen

Kapitel 1

Einleitung

Die Implementierung eines geeigneten und zielgerichteten Projektmanagements stellt für viele Unternehmen im Hinblick auf kürzer werdende Innovationszyklen und sich verändernde Marktanforderungen einen immer wichtigeren Erfolgsfaktor dar. Eine entscheidende Bedeutung kommt dabei vor allem der Projektplanung als Bindeglied zwischen der Vorbereitungs- und der Ausführungsphase eines Projekts zu. Die Aufgabe der Projektplanung besteht grundsätzlich darin, die Tätigkeiten (Vorgänge) eines Projekts unter Berücksichtigung der Zeitbeziehungen zwischen den Vorgängen sowie eventuell bestehender Ressourcenrestriktionen so einzuplanen, dass eine vorgegebene Zielsetzung bestmöglich erreicht wird. Die Projektplanung mit Ressourcenrestriktionen wird dabei als ressourcenbeschränkte Projektplanung bezeichnet. Da der Erfolg vieler Unternehmen von einem effizienten und kostengünstigen Einsatz begrenzt verfügbarer Ressourcen abhängt, können geeignete Modelle und quantitative Methoden für die ressourcenbeschränkte Projektplanung ein entscheidendes Wettbewerbspotential bieten.

In den letzten Jahrzehnten wurden unterschiedliche Modelle zur ressourcenbeschränkten Projektplanung entwickelt, die sich insbesondere darin unterscheiden, von welcher Zielsetzung und von welchen vereinfachenden Annahmen ausgegangen wird. In den meisten dieser Modelle werden zur Abbildung von Kapazitätsrestriktionen erneuerbare Ressourcen betrachtet, die dadurch gekennzeichnet sind, dass sie in jeder Zeitperiode in einer bestimmten Menge zur Verfügung stehen[1] und durch die Vorgänge des Projekts während ihrer Ausführung belegt werden. Erneuerbare Ressourcen werden beispielsweise zur Modellierung der Verfügbarkeit von Arbeitskräften oder Maschinen eingesetzt. Ein entscheidender Nachteil der erneuerbaren Ressourcen besteht jedoch darin, dass bereits vor der

[1] Es wird zur Vereinfachung der Darstellungen zunächst davon ausgegangen, dass sich die Kapazität einer erneuerbaren Ressource ausschließlich auf Zeitperioden bezieht. Es sei allerdings darauf verwiesen, dass auch Modelle mit erneuerbaren Ressourcen in der Literatur behandelt werden, deren Kapazitäten für jeden Zeitpunkt des Planungshorizonts definiert sind (vgl. z. B. Neumann und Schwindt, 2003).

Einplanung der Vorgänge des Projekts für jede Zeitperiode festgelegt werden muss, wie viele Mengeneinheiten von jeder Ressource zur Verfügung stehen. Diese Voraussetzung führt allerdings dazu, dass praxisrelevante Restriktionen, wie Arbeitszeitvereinbarungen oder Vorgaben zur Höchstauslastung von Maschinen über längere Zeiträume, durch Modelle mit erneuerbaren Ressourcen nicht abgebildet werden können. Als Beispiel sei eine Arbeitszeitvereinbarung betrachtet, die vorgibt, dass ein Mitarbeiter in jeder Woche von Montag bis Samstag an fünf Tagen arbeitet, wobei der Ruhetag des Mitarbeiters flexibel gehalten werden soll. Durch eine Modellierung mit erneuerbaren Ressourcen könnte der Ruhetag des Mitarbeiters lediglich im Voraus durch eine erneuerbare Ressource mit zeitlich variierender Kapazität festgelegt werden, wohingegen die geforderte Flexibilität nicht darstellbar wäre. Eine Möglichkeit, um Restriktionen wie die Arbeitszeitvereinbarung in Modellen der ressourcenbeschränkten Projektplanung abzubilden, bieten partiell erneuerbare Ressourcen, mit deren Hilfe Kapazitätsrestriktionen auch über mehrere Zeitperioden modelliert werden können. Der grundsätzliche Unterschied zu den erneuerbaren Ressourcen besteht darin, dass erst durch die Einplanung der Vorgänge des Projekts festgelegt wird, welche verfügbare Ressourceneinheit in welcher Zeitperiode zur Ausführung der Vorgänge eingesetzt wird. Für die Arbeitszeitvereinbarung könnte beispielsweise eine partiell erneuerbare Ressource mit einer Kapazität von fünf Einheiten über alle Wochentage von Montag bis Samstag verwendet werden, wobei sich die Arbeitstage des Mitarbeiters erst aus der Einplanung der Vorgänge ergeben würden.

Eine weitere vereinfachende Annahme, die in vielen Modellen der ressourcenbeschränkten Projektplanung vorausgesetzt wird, geht davon aus, dass ausschließlich Vorrangbeziehungen zwischen den Vorgangspaaren eines Projekts vorliegen. Durch diese Art der Zeitbeziehung wird impliziert, dass ein Vorgang erst dann starten kann, sobald ein anderer Vorgang endet. Da diese vereinfachende Annahme allerdings viele praktische Anwendungen ausschließt, wurden in der Literatur Modelle mit zeitlichen Mindest- und Höchstabständen (allgemeine Zeitbeziehungen) eingeführt, die es beispielsweise ermöglichen, Abschlusstermine für Teilprojekte vorzugeben oder technologisch bedingte Zeitfenster für die Ausführung von Fertigungsprozessen festzulegen.

Gegenstand der vorliegenden Arbeit ist das Projektdauerminimierungsproblem mit allgemeinen Zeitbeziehungen und partiell erneuerbaren Ressourcen (RCPSP/max-π). Für das entsprechende ressourcenbeschränkte Projektplanungsproblem sind die Vorgänge unter Berücksichtigung der vorgegebenen Zeit- und Ressourcenrestriktionen so einzuplanen, dass das Projekt frühestmöglich beendet wird. Das RCPSP/max-π stellt eine Verallgemeinerung des klassischen Projektdauerminimierungsproblems mit Vorrangbeziehungen und erneuerbaren Ressourcen (RCPSP) dar und wurde erstmals in den Arbeiten Wa-

termeyer und Zimmermann (2018, 2020), die im Rahmen dieser Dissertation entstanden sind, eingeführt und untersucht. Auf die Inhalte der genannten Veröffentlichungen, die zum Teil in der vorliegenden Dissertation übernommen wurden, wird an geeigneter Stelle dieser Arbeit verwiesen.

Das Ziel der vorliegenden Arbeit besteht darin, Branch-and-Bound-Verfahren für das RCPSP/max-π zu entwickeln und deren Leistungsfähigkeit anhand geeigneter Testinstanzen zu bewerten. Weiterhin soll untersucht werden, welcher Zusammenhang zwischen partiell erneuerbaren Ressourcen und anderen Modellierungskonzepten der Projektplanung besteht, wodurch ein Bezug des RCPSP/max-π zu anderen Modellen der Projektplanung hergestellt werden soll.

Die Arbeit ist wie folgt gegliedert:

In **Kapitel 2** wird zunächst das Projektdauerminimierungsproblem mit allgemeinen Zeitbeziehungen und partiell erneuerbaren Ressourcen formalisiert sowie auf Bezeichner und Konventionen eingegangen, die im weiteren Verlauf der vorliegenden Arbeit verwendet werden. Zur Verdeutlichung der Relevanz der partiell erneuerbaren Ressourcen für die Modellierung realer Anwendungen wird des Weiteren ein Beispielprojekt betrachtet sowie auf die bislang veröffentlichte Literatur zu partiell erneuerbaren Ressourcen eingegangen.

Aufbauend auf den Erkenntnissen aus der Literatur, dass die partiell erneuerbaren Ressourcen die erneuerbaren und nicht-erneuerbaren Ressourcen verallgemeinern[2], wird in **Kapitel 3** gezeigt, dass auch weitere Konzepte der Projektplanung, wie z. B. zeitliche Höchstabstände, durch partiell erneuerbare Ressourcen (sowie durch zusätzliche Vorgänge und Zeitbeziehungen) modelliert werden können. Anschließend wird auf die Lösbarkeit der unterschiedlichen Projektdauerminimierungsprobleme, die aus den betrachteten Modellierungskonzepten des Kapitels hervorgehen, durch exakte Verfahren für Projektdauerminimierungsprobleme mit partiell erneuerbaren Ressourcen eingegangen. Den Abschluss des Kapitels stellt schließlich eine Komplexitätsbetrachtung für die in der vorliegenden Arbeit untersuchte Problemstellung dar.

Kapitel 4 führt zunächst ein Zeitplanungsproblem mit eingeschränkten Startzeitpunkten der Vorgänge eines Projekts ein, das jedem Enumerationsknoten der in dieser Arbeit entwickelten Branch-and-Bound-Verfahren zugeordnet ist. Für diese Problemstellung werden verschiedene Zeitplanungsverfahren vorgestellt, die sowohl zur Lösung des Planungs-

[2] Die partiell erneuerbaren Ressourcen stellen nur dann eine Verallgemeinerung der erneuerbaren und nicht-erneuerbaren Ressourcen (sowie aller weiteren betrachteten Modellierungskonzepte in Kapitel 3) dar, wenn angenommen wird, dass alle Vorgänge des Projekts nur zu ganzzahligen Zeitpunkten starten können und eine maximale Projektdauer vorgegeben ist.

problems als auch zur Ausführung von Konsistenztests eingesetzt werden. Basierend auf den Zeitplanungsverfahren werden schließlich verschiedene Konsistenztests entwickelt, die zur Eingrenzung der Wertebereiche der Startzeitpunkte aller Vorgänge des Projekts in jedem Enumerationsknoten ausgeführt werden können. Abschließend wird ein Verfahren zur Bestimmung einer destruktiven unteren Schranke für die kürzeste Projektdauer über alle zulässigen Lösungen des Zeitplanungsproblems in jedem Knoten beschrieben.

In **Kapitel 5** werden zwei relaxationsbasierte und ein konstruktionsbasiertes Branch-and-Bound-Verfahren zur Lösung des Projektdauerminimierungsproblems mit allgemeinen Zeitbeziehungen und partiell erneuerbaren Ressourcen entwickelt. Für jedes der Lösungsverfahren wird zunächst das zugehörige Enumerationsschema herausgearbeitet, das anschließend um untere Schrankenwerte und Konsistenztests zu einem Branch-and-Bound-Verfahren erweitert wird. Ferner werden für zwei der Enumerationsansätze Verfahren zur Redundanzvermeidung vorgestellt und dem jeweiligen Branch-and-Bound-Verfahren hinzugefügt.

Zur Einordnung der Leistungsfähigkeit der entwickelten Branch-and-Bound-Verfahren im Vergleich zu einem MILP-Solver werden in **Kapitel 6** ganzzahlige lineare Programme für das RCPSP/max-π entwickelt, die auf zeitindexbasierten Modellformulierungen aus der Literatur basieren.

In **Kapitel 7** werden die Ergebnisse einer umfassenden experimentellen Performance-Analyse diskutiert. Dazu werden die in Kapitel 5 entwickelten Branch-and-Bound-Verfahren zunächst einander gegenübergestellt und anschließend mit einem weiteren exakten Lösungsverfahren aus der Literatur verglichen. Das Branch-and-Bound-Verfahren mit der besten Performance wird schließlich einem MILP-Solver, der zur Lösung der in Kapitel 6 vorgestellten ganzzahligen linearen Programme eingesetzt wird, gegenübergestellt und mit den besten bislang bekannten heuristischen Verfahren für Projektplanungsprobleme mit partiell erneuerbaren Ressourcen verglichen.

Abschließend werden in **Kapitel 8** die wichtigsten Ergebnisse der vorliegenden Arbeit zusammengefasst, und es erfolgt ein Ausblick auf zukünftige Themenbereiche und Forschungsfragen.

Kapitel 2

Einführung in die Problemstellung

Das Projektdauerminimierungsproblem mit partiell erneuerbaren Ressourcen und allgemeinen Zeitbeziehungen (RCPSP/max-π) verbindet zwei Erweiterungen des klassischen RCPSP, die bereits im Rahmen des RCPSP mit partiell erneuerbaren Ressourcen (RCPSP/π) sowie mit allgemeinen Zeitbeziehungen (RCPSP/max) getrennt voneinander untersucht wurden. Das RCPSP/max-π leistet dementsprechend einen wichtigen Beitrag dazu, dass weitere reale Anwendungen durch Modelle der ressourcenbeschränkten Projektplanung abgebildet werden können.

Im ersten Abschnitt dieses Kapitels wird zunächst ein mathematisches Modell für das RCPSP/max-π entwickelt. In diesem Zusammenhang werden unter anderem die wichtigsten Bezeichner und Konventionen für das RCPSP/max-π eingeführt. In Abschnitt 2.2 werden unterschiedliche Möglichkeiten der Modellierung mit Hilfe von partiell erneuerbaren Ressourcen anhand eines Anwendungsbeispiels herausgestellt. Im letzten Abschnitt 2.3 werden schließlich die wichtigsten Veröffentlichungen, die in den letzten Jahrzehnten zu partiell erneuerbaren Ressourcen publiziert wurden, in den Kontext dieser Arbeit gestellt.

2.1 Projektplanung mit partiell erneuerbaren Ressourcen

Das RCPSP/max-π umfasst die Einplanung einer Menge $V := \{0, 1, \ldots, n+1\}$ von Vorgängen unter der Einhaltung vorgegebener Zeit- und Ressourcenrestriktionen. Die Zielsetzung der Problemstellung ist die Minimierung des Fertigstellungszeitpunkts des Projekts. Jedem Vorgang $i \in V$ ist eine nicht-unterbrechbare Ausführungsdauer $p_i \in \mathbb{Z}_{\geq 0}$ zugeordnet. Die Vorgangsmenge V unterteilt sich in reale Vorgänge $i \in V^r$ mit $p_i > 0$ und fiktive Vorgänge (Ereignisse) $i \in V^e$ mit $p_i = 0$. Die fiktiven Vorgänge 0 und $n+1$ repräsentieren

jeweils den Projektbeginn und das Projektende. Weitere fiktive Vorgänge können allgemeine Etappenziele wie beispielsweise den Abschluss eines Teilprojekts modellieren, die auch als Meilensteine bezeichnet werden. Durch einen Schedule $S := (S_i)_{i \in V}$ wird jedem Vorgang $i \in V$ ein Startzeitpunkt $S_i \in \mathbb{Z}_{\geq 0}$ zugeordnet. Jedes Projekt startet zum Zeitpunkt 0 ($S_0 = 0$) und wird spätestens nach $\bar{d} \in \mathbb{Z}_{\geq 0}$ Zeiteinheiten beendet ($S_{n+1} \leq \bar{d}$). $\bar{d}$ entspricht dabei dem spätesten Zeitpunkt des Planungshorizonts $\mathcal{H} := \{0, 1, \ldots, \bar{d}\}$ bzw. einer oberen Schranke für den Fertigstellungszeitpunkt des Projekts. Zwischen den Vorgängen des Projekts sind Zeitbeziehungen zu berücksichtigen, die sich aus technologischen oder ablauforganisatorischen Gründen ergeben können. Eine Zeitbeziehung zwischen zwei Vorgängen $i, j \in V$, $i \neq j$ wird durch die Bedingung

$$S_j \geq S_i + \delta_{ij}$$

mit $\delta_{ij} \in \mathbb{Z}$ als Zeitabstand zwischen den Startzeitpunkten der Vorgänge beschrieben. Dabei werden sowohl zeitliche Mindestabstände ($\delta_{ij} \geq 0$) als auch zeitliche Höchstabstände ($\delta_{ij} < 0$) betrachtet. Es ist zu beachten, dass die Modellierung der Zeitbeziehungen durch die Startzeitpunkte der Vorgänge keine Beschränkung der Allgemeinheit darstellt, da sich aufgrund der deterministischen Zeitabstände und Ausführungsdauern jede andere Anordnungsbeziehung in eine Start-Start-Anordnungsbeziehung überführen lässt (vgl. Bartusch et al., 1988). Im Folgenden beschreibt $E \subset V \times V$ die Menge aller Vorgangspaare (i, j), für die eine Zeitbeziehung vorliegt. Das Projektdauerminimierungsproblem unter Zeitrestriktionen (PS) lässt sich durch das folgende mathematische Modell beschreiben:

$$\left.\begin{array}{lll} \text{Minimiere} & S_{n+1} & \\ \text{u. d. N.} & S_j - S_i \geq \delta_{ij} & ((i,j) \in E) \\ & S_0 = 0 & \\ & S_i \in \mathbb{Z}_{\geq 0} & (i \in V) \end{array}\right\} \text{(PS)}$$

Zur Vereinfachung der Modellierung von (PS) wird vorausgesetzt, dass eine Zeitbeziehung zwischen dem Projektende $n + 1$ und dem Projektstart 0 mit einem Zeitabstand $\delta_{n+1,0} = -\bar{d}$ vorliegt ($(n + 1, 0) \in E$). Entsprechend erfüllt jede zulässige Lösung S des Problems (PS) die Bedingung $S_{n+1} \leq \bar{d}$. Weiterhin wird angenommen, dass die Bedingungen $S_i \geq 0$ und $S_i + p_i \leq S_{n+1}$ für alle Vorgänge $i \in V$ durch die gegebenen Zeitbeziehungen erfüllt sind, sodass sich kein Vorgang $i \in V$ vor dem Projektbeginn ($S_0 = 0$) oder nach dem Projektende (S_{n+1}) in Ausführung befindet. Im Folgenden wird der zulässige Bereich

$$\mathcal{S}_T := \{S \in \mathbb{Z}_{\geq 0}^{n+2} \mid S_j - S_i \geq \delta_{ij} \text{ für alle } (i,j) \in E \wedge S_0 = 0\}$$

des Problems (PS) als Menge aller zeitzulässigen Schedules (oder als zeitzulässiger Bereich) bezeichnet und jeder Schedule $S \in \mathcal{S}_T$ zeitzulässig genannt. Durch die Modellierung des Problems (PS) als Vorgangsknotennetzplan können Algorithmen der Graphentheorie für die Zeitplanung eingesetzt werden (vgl. z. B. Zimmermann, 2001, Abschnitt 1.1 bzw. Zimmermann et al., 2010, Abschnitt 1.4.3). Im Folgenden soll unter der Zeitplanung eines Projekts die Bestimmung der frühesten und spätesten Startzeitpunkte der Vorgänge sowie die Ermittlung der (indirekten) zeitlichen Mindest- und Höchstabstände zwischen allen Vorgangspaaren verstanden werden. Das Problem (PS) wird als Vorgangsknotennetzplan $N = (V, E, \delta)$ mit Knotenmenge V, Pfeilmenge E und Pfeilbewertung $\delta := (\delta_{ij})_{(i,j)\in E}$ modelliert. Zur Bestimmung der frühesten und spätesten zeitzulässigen Startzeitpunkte ES_i und LS_i aller Vorgänge $i \in V$ können Label-Correcting-Verfahren mit Zeitkomplexitäten von jeweils $\mathcal{O}(|V||E|)$ verwendet werden (vgl. Neumann et al., 2003, Abschnitt 1.3). Die Schedules $ES := (ES_i)_{i\in V}$ und $LS := (LS_i)_{i\in V}$ entsprechen dem eindeutigen Minimal- bzw. Maximalpunkt des zeitzulässigen Bereichs $\mathcal{S}_T$ und werden im Folgenden als früheste und späteste zeitzulässige Schedules bezeichnet. Ein Schedule $S \in \mathcal{M} \subseteq \mathbb{Z}^{n+2}$ wird dabei als Minimalpunkt einer Menge $\mathcal{M}$ bezeichnet, wenn kein anderer Schedule $S' \in \mathcal{M}$ mit $S'_i \leq S_i$ für alle Vorgänge $i \in V$ ($S' \leq S$) existiert. Analog dazu wird ein Schedule $S \in \mathcal{M}$ Maximalpunkt einer Menge $\mathcal{M}$ genannt, falls kein anderer Schedule $S' \in \mathcal{M}$ mit $S' \geq S$ existiert. Die (indirekten) Zeitabstände d_{ij} zwischen den Startzeitpunkten aller Vorgangspaare $(i, j) \in V \times V$ können mit Hilfe des Floyd-Warshall-Algorithmus mit einer Zeitkomplexität von $\mathcal{O}(|V|^3)$ berechnet werden (vgl. Neumann und Morlock, 2002, Abschnitt 2.4.4). Der Zeitabstand d_{ij} entspricht dabei der Länge eines längsten Wegs von Vorgang i zu Vorgang j in Netzplan N. Es ist zu beachten, dass für die frühesten und spätesten zeitzulässigen Startzeitpunkte ES_i und LS_i der Vorgänge $i \in V$ die Zusammenhänge $ES_i = d_{0i}$ und $LS_i = -d_{i0}$ gelten, sodass die Schedules ES und LS auch durch den Floyd-Warshall-Algorithmus berechnet werden können. Im weiteren Verlauf dieser Arbeit wird die Matrix $D = (d_{ij})_{i,j\in V}$ der Längen längster Wege in Netzplan N zwischen allen Vorgangspaaren des Projekts als Distanzmatrix bezeichnet.

Das RCPSP/max-π betrachtet partiell erneuerbare Ressourcen, die dadurch gekennzeichnet sind, dass sie nur auf vorgegebenen Zeitperioden des Planungshorizonts durch die Ausführung von Vorgängen in Anspruch genommen werden. Jeder partiell erneuerbaren Ressource $k \in \mathcal{R}$ ist eine Ressourcenkapazität $R_k \in \mathbb{Z}_{\geq 0}$ sowie eine Teilmenge $\Pi_k \subseteq \{1, 2, \ldots, \bar{d}\}$ aller Perioden des Planungshorizonts zugeordnet. In jeder Periode der Menge Π_k, in der sich ein Vorgang $i \in V$ in Ausführung befindet, werden $r^d_{ik} \in \mathbb{Z}_{\geq 0}$ Einheiten der Ressource k durch Vorgang i in Anspruch genommen. Im weiteren Verlauf wird r^d_{ik} als Bedarf von Vorgang i an Ressource k bezeichnet. Die Anzahl der Perioden

der Menge Π_k, in denen sich Vorgang i mit Startzeitpunkt S_i in Ausführung befindet, wird durch die Ressourcenbelegung $r^u_{ik}(S_i) := |]S_i, S_i + p_i] \cap \Pi_k|$ beschrieben, woraus die Ressourceninanspruchnahme $r^c_{ik}(S_i) := r^u_{ik}(S_i)\, r^d_{ik}$ folgt.

Die Abhängigkeit der Ressourcenbelegung (Ressourceninanspruchnahme) vom Startzeitpunkt eines Vorgangs wird in Abbildung 2.1 veranschaulicht. Das Beispiel zeigt den Belegungsverlauf einer Ressource k mit $\Pi_k = \{3, 4, 5, 6, 7, 12\}$ in Abhängigkeit vom Startzeitpunkt S_i eines Vorgangs i mit einer Ausführungsdauer $p_i = 3$ und einem Ressourcenbedarf $r^d_{ik} = 1$. Die Menge Π_k sowie die Ressourcenbelegung $r^u_{ik}(0) = 1$ von Vorgang i mit Startzeitpunkt $S_i = 0$ werden durch schraffierte Flächen verdeutlicht. Der untere Teil der Abbildung zeigt schließlich den Verlauf der Ressourcenbelegung $r^u_{ik}(S_i)$ über alle Startzeitpunkte $S_i \in \{0, 1, \ldots, 12\}$ des Vorgangs.

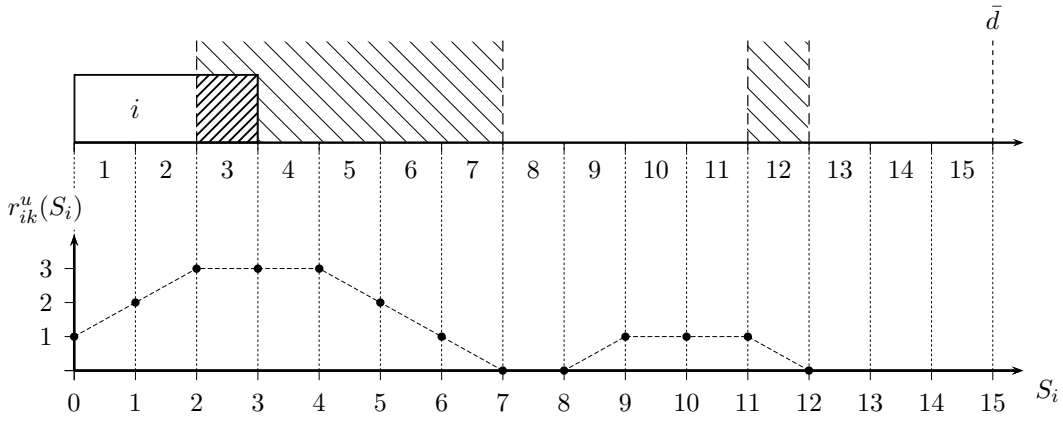

Abbildung 2.1: Abhängigkeit der Ressourcenbelegung vom Startzeitpunkt

Die Gesamtinanspruchnahme $r^c_k(S)$ einer Ressource $k \in \mathcal{R}$ für einen gegebenen Schedule S ergibt sich aus der Summe der Ressourceninanspruchnahmen $r^c_{ik}(S_i)$ aller Vorgänge des Projekts. Entsprechend können die Ressourcenrestriktionen für das RCPSP/max-π durch

$$r^c_k(S) := \sum_{i \in V} r^c_{ik}(S_i) \leq R_k \qquad (k \in \mathcal{R})$$

beschrieben werden. Das mathematische Modell für das RCPSP/max-π kann wie folgt angegeben werden:

$$\left.\begin{array}{lll} \text{Minimiere} & S_{n+1} & \\ \text{u. d. N.} & r^c_k(S) \leq R_k & (k \in \mathcal{R}) \\ & S_j - S_i \geq \delta_{ij} & ((i,j) \in E) \\ & S_0 = 0 & \\ & S_i \in \mathbb{Z}_{\geq 0} & (i \in V) \end{array}\right\} \text{(P)}$$

Im weiteren Verlauf dieser Arbeit wird

$$\mathcal{S}_R := \{S \in \mathbb{Z}_{\geq 0}^{n+2} \mid r_k^c(S) \leq R_k \text{ für alle } k \in \mathcal{R}\}$$

als Menge aller ressourcenzulässigen Schedules (oder als ressourcenzulässiger Bereich) bezeichnet und jeder Schedule $S \in \mathcal{S}_R$ ressourcenzulässig genannt. Weiterhin wird jeder zeit- und ressourcenzulässige Schedule $S \in \mathcal{S}_T \cap \mathcal{S}_R$ als zulässig bezeichnet, sodass $\mathcal{S} := \mathcal{S}_T \cap \mathcal{S}_R$ der Menge aller zulässigen Schedules entspricht. Die Menge aller Schedules, die das Problem (P) optimal lösen, wird zudem durch $\mathcal{OS}$ angegeben.

2.2 Anwendungsbeispiel

Im Folgenden soll die Modellierung eines Projektplanungsproblems mit partiell erneuerbaren Ressourcen sowie mit allgemeinen Zeitbeziehungen anhand eines Anwendungsbeispiels aus dem Bereich der Softwareentwicklung motiviert werden. Das Beispiel betrachtet ein Projekt eines Softwareunternehmens, das mit der Entwicklung einer Anwendungssoftware für einen Kunden beauftragt wurde. Der Auftrag umfasst die Entwicklung und die Systemintegration der Software und soll innerhalb von zwei Wochen abgeschlossen sein. Das Softwareunternehmen stellt für das Projekt einen Kundenbetreuer (K) und einen Programmierer (P) ab. Weiterhin wird ein bestimmtes Datenvolumen (D) für den Zugriff auf Daten des Kunden sowie für den Datenaustausch zwischen dem Kunden und dem Softwareunternehmen benötigt. Für den Kundenbetreuer und den Programmierer sind unterschiedliche Arbeitszeitvereinbarungen zu berücksichtigen. Der Kundenbetreuer arbeitet höchstens fünf Tage innerhalb einer Woche und wird für ein Wochenende freigestellt, falls er am Wochenende zuvor sowohl am Samstag als auch am Sonntag eingesetzt wurde. Der Programmierer dagegen arbeitet nicht mehr als acht Tage und höchstens zwei Wochenendtage innerhalb von zwei aufeinanderfolgenden Wochen. Das Projekt benötigt insgesamt ein Datenvolumen von 25 GB. Da das Softwareunternehmen am Wochenende weniger für die Datenübertragung bezahlen muss, sollen vom gesamten Datenvolumen mindestens 40 % am Wochenende verbraucht werden. Für die Planung des Projekts wird angenommen, dass eine Diskretisierung des Planungshorizonts auf Tagesbasis ausreichend ist. Entsprechend wird jeder Wochentag genau durch eine Zeitperiode modelliert, wobei die erste Periode einem Samstag entspricht (s. Abbildung 2.3).

In Tabelle 2.1 sind alle partiell erneuerbaren Ressourcen $\mathcal{R} = \{1, 2, \ldots, 35\}$ aufgelistet, die für die Modellierung der beschriebenen Restriktionen erforderlich sind. Um den Bezug der einzelnen partiell erneuerbaren Ressourcen zum Kundenbetreuer (K), zum

			Π_k	R_k		
Π_1	Π_{19}	–	$\{1\}$	1	1	–
Π_2	Π_{20}	–	$\{2\}$	1	1	–
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Π_{14}	Π_{32}	–	$\{14\}$	1	1	–
Π_{15}	–	–	$\{1, 2, \ldots, 7\}$	5	–	–
Π_{16}	–	–	$\{8, 9, \ldots, 14\}$	5	–	–
Π_{17}	–	–	$\{1, 2, 8\}$	2	–	–
Π_{18}	–	–	$\{1, 2, 9\}$	2	–	–
–	Π_{33}	–	$\{1, 2, 8, 9\}$	–	2	–
–	Π_{34}	–	$\{1, 2, \ldots, 14\}$	–	8	–
–	–	Π_{35}	$\{3, \ldots, 7, 10, \ldots, 14\}$	–	–	15

Tabelle 2.1: Ressourcen des Beispielprojekts

Programmierer (P) und zum Datenvolumen (D) zu verdeutlichen, werden im Folgenden die Mengen $\widetilde{\mathcal{R}}_{\mathrm{K}} = \{1, 2, \ldots, 18\}$, $\widetilde{\mathcal{R}}_{\mathrm{P}} = \{19, 20, \ldots, 34\}$ und $\widetilde{\mathcal{R}}_{\mathrm{D}} = \{35\}$ betrachtet. Für jede Ressource $k \in \mathcal{R}$ sind die Periodenmenge Π_k und die Kapazität R_k in einer Zeile von Tabelle 2.1 aufgeführt. Beispielsweise können aus der ersten Zeile $\Pi_1 = \Pi_{19} = \{1\}$ und $R_1 = R_{19} = 1$ mit $k = 1 \in \widetilde{\mathcal{R}}_{\mathrm{K}}$ und $k = 19 \in \widetilde{\mathcal{R}}_{\mathrm{P}}$ entnommen werden. Die Restriktionen, die für den Kundenbetreuer (K), den Programmierer (P) und das Datenvolumen (D) einzuhalten sind, werden jeweils durch die Ressourcen $k = 15, 16, 17, 18$, $k = 33, 34$ und $k = 35$ modelliert. Beispielsweise stellen die Ressourcen $k = 17, 18$ sicher, dass der Kundenbetreuer an einem Wochenende freigestellt wird, falls er an dem direkt davorliegenden Wochenende sowohl am Samstag als auch am Sonntag gearbeitet hat. Alle weiteren Ressourcen ($k = 1, 2, \ldots, 14, 19, 20, \ldots, 32$) sorgen dafür, dass jede Arbeitskraft (K, P) an jedem Tag höchstens einen Vorgang bearbeitet. Dabei ist zu beachten, dass diese Restriktionen auch durch erneuerbare Ressourcen modellierbar wären, worauf in Abschnitt 3.1 noch näher eingegangen wird.

In Tabelle 2.2 sind alle Vorgänge, die im Verlauf des Projekts bearbeitet werden müssen, aufgeführt. Jedem Vorgang ist eine Ausführungsdauer p_i sowie ein Ressourcenbedarf an der Ressource $\sigma \in \{\mathrm{K}, \mathrm{P}, \mathrm{D}\}$ zugeordnet. Zur Vereinfachung wird die Notation $\tilde{r}^d_{i\sigma} = a$ verwendet, die einen Bedarf $r^d_{ik} = a$ von Vorgang i für alle Ressourcen $k \in \widetilde{\mathcal{R}}_\sigma$ impliziert. Beispielsweise hat Vorgang $i = 3$ eine Ausführungsdauer von $p_3 = 3$, einen Bedarf $r^d_{3k} = 1$ für alle Ressourcen $k \in \widetilde{\mathcal{R}}_{\mathrm{P}}$ und einen Bedarf $r^d_{3k} = 5$ für alle Ressourcen $k \in \widetilde{\mathcal{R}}_{\mathrm{D}}$.

Der Projektnetzplan in Abbildung 2.2 zeigt unter anderem die Zeitbeziehungen zwischen den Vorgängen des Projekts. Das Projekt beginnt zunächst mit einem Beratungsgespräch

$i \in V$	Beschreibung	p_i	$\widetilde{r}^d_{i\mathrm{K}}$	$\widetilde{r}^d_{i\mathrm{P}}$	$\widetilde{r}^d_{i\mathrm{D}}$
0	Projektstart	0	0	0	0
1	Kundenberatung	3	1	0	0
2	Besprechung der Implementierung	1	1	1	0
3	Datenzugriff und Implementierung	3	0	1	5
4	Automatisierter Testlauf	2	0	0	4
5	Systemintegration vor Ort	1	1	1	2
6	Schulung der Mitarbeiter	1	1	0	0
7	Projektende	0	0	0	0

Tabelle 2.2: Vorgänge des Beispielprojekts

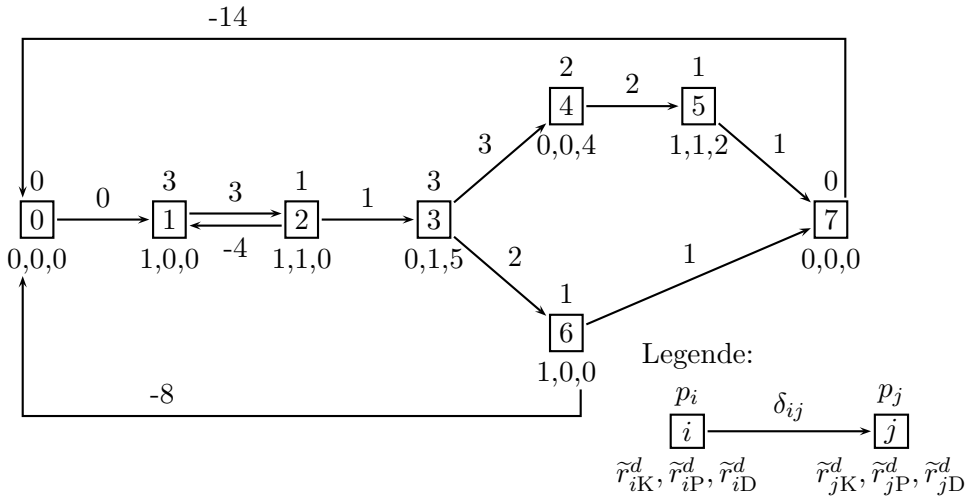

Abbildung 2.2: Projektnetzplan des Beispielprojekts

des Kundenbetreuers mit dem Kunden, um die Anforderungen der Software für das Unternehmen zu klären ($i = 1$). Im Anschluss daran ($\delta_{12} = 3$) folgt eine Besprechung des Kundenbetreuers mit dem Programmierer ($i = 2$), in der die Ausgestaltung der Software festgelegt wird. Um sicherzustellen, dass der Kundenbetreuer sich noch an möglichst viele Details der Beratung mit dem Kunden erinnern kann, soll die Besprechung mit dem Programmierer spätestens einen Tag nach dem Ende von Vorgang $i = 1$ stattfinden ($\delta_{21} = -4$). Nach Abschluss des Vorgangs $i = 2$ ($\delta_{23} = 1$) kann schließlich die Softwareanwendung durch den Programmierer erstellt werden, wobei ein Zugriff auf die Daten des Kunden erforderlich ist ($i = 3$). Während der automatisierte Testlauf ($i = 4$) zur Überprüfung aller Funktionen der Softwareanwendung nicht vor dem Ende von Vorgang $i = 3$ gestartet wird ($\delta_{34} = 3$), kann die Schulung der Mitarbeiter des Kunden ($i = 6$) bereits einen Tag vor dem Ende von Vorgang $i = 3$ stattfinden ($\delta_{36} = 2$). Um für Mitarbeiter mit Schwierigkeiten im Umgang mit der Softwareanwendung weitergehende Schulungen

anbieten zu können, soll die geplante Schulung ($i = 6$) spätestens am Ende des zweiten Wochenendes abgeschlossen sein ($\delta_{60} = -8$). Den letzten verbleibenden Arbeitsschritt des Projekts stellt die Systemintegration sowie die Behebung möglicher Fehler der Softwareanwendung vor Ort ($i = 5$) dar, der erst nach Abschluss des Testlaufs gestartet werden kann ($\delta_{45} = 2$).

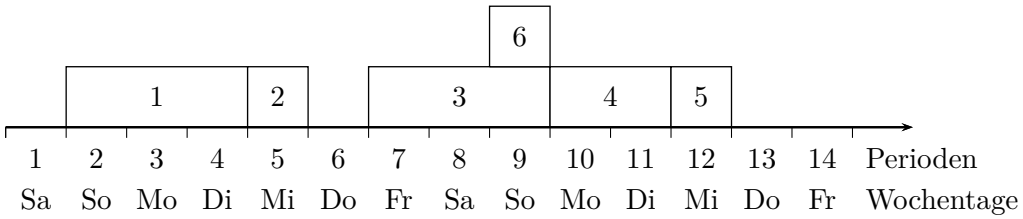

Abbildung 2.3: Optimaler Schedule für das Beispielprojekt

Abbildung 2.3 zeigt einen optimalen Schedule $S^* = (0, 1, 4, 6, 9, 11, 8, 12)$ für das Beispielprojekt mit einer (minimalen) Projektdauer $S_7^* = 12$. Es sollte beachtet werden, dass Vorgang $i = 3$ durch die Vorgabe, dass mindestens 40 % des gesamten benötigten Datenvolumens am Wochenende zu nutzen ist, über das gesamte zweite Wochenende ausgeführt werden muss. Aufgrund des einzuhaltenden Mindestabstands $\delta_{36} = 2$ kann die Schulung der Mitarbeiter ($i = 6$) schließlich nur noch am Sonntag stattfinden, wodurch Vorgang $i = 1$ nicht über das gesamte erste Wochenende ausgeführt werden darf (s. Wochenend-Restriktion des Kundenbetreuers $k = 17, 18$).

2.3 Literaturüberblick

In den letzten Jahrzehnten wurden die partiell erneuerbaren Ressourcen für die Modellierung von Problemstellungen in unterschiedlichen Anwendungsbereichen verwendet. In Drexl et al. (1993) wurden die partiell erneuerbaren Ressourcen erstmalig eingeführt und für die Vorlesungsplanung an Universitäten eingesetzt. Unter anderem zeigen die Autoren, dass sich Restriktionen wie Raumkapazitäten oder Verfügbarkeiten von Lehrpersonal durch partiell erneuerbare Ressourcen modellieren lassen. Eine ähnliche Problemstellung wird in Drexl und Salewski (1997) mit einer Stundenplanerstellung in Schulen behandelt, in der die partiell erneuerbaren Ressourcen zusätzlich für die Modellierung von Bedingungen wie gleichzeitig stattfindenden Unterrichtsstunden verwendet werden. In der Arbeit von Bartsch et al. (2006) werden die partiell erneuerbaren Ressourcen für die Spielplanerstellung der ersten Fußballbundesligen in Deutschland und Österreich eingesetzt. Unter anderem werden Restriktionen wie eine maximale Anzahl an Spielen innerhalb einer

bestimmten Region sowie Verfügbarkeiten von Stadien durch die partiell erneuerbaren Ressourcen modelliert.

In Knust (2010) wird die Spielplanerstellung für eine Amateur-Tischtennisliga durch ein Mehr-Modus-Projektplanungsproblem mit partiell erneuerbaren und erneuerbaren Ressourcen mit zeitlich variierenden Kapazitäten modelliert. Für die Problemstellung werden Vorgänge mit Ausführungsdauern von jeweils einer Zeiteinheit sowie nur auf zusammenhängenden Zeitperioden definierte partiell erneuerbare Ressourcen betrachtet. Die Zielsetzung entspricht einer zu minimierenden Bestrafungsfunktion nicht wünschenswerter Eigenschaften des Spielplans. Es wird gezeigt, dass unter den angenommenen Bedingungen ein serielles Generierungsschema basierend auf Aktivitätslisten verwendet werden kann, um eine zulässige Lösung bei vorgegebener Moduszuweisung zu bestimmen. Zur Ermittlung einer Näherungslösung wird ein hierarchischer Ansatz eingesetzt, der zunächst eine Moduszuweisung vornimmt und darauffolgend den genetischen Algorithmus aus Hartmann (1998) zur Bestimmung eines Schedules anwendet.

In Briskorn und Fliedner (2012) wird gezeigt, dass die in der Arbeit behandelte Problemstellung, die sich an eine reale Anwendung zur Containerabfertigung in Umschlagsbahnhöfen anlehnt (vgl. Kellner et al., 2012), einem Spezialfall des Zulässigkeitsproblems des RCPSP/π entspricht. Die Problemstellung zur Containerabfertigung kann durch das RCPSP/π modelliert werden, indem die Periodenmengen der Ressourcen des Projekts gleich viele aufeinanderfolgende Perioden umfassen, paarweise disjunkt sind und den gesamten Planungshorizont überdecken. Weiterhin haben alle Ressourcen die gleiche Kapazität und alle Vorgänge einen identischen Bedarf an jeder Ressource. Unter diesen Voraussetzungen ergibt sich aus den Untersuchungen in Briskorn und Fliedner (2012), dass das Zulässigkeitsproblem des RCPSP/π für $|\mathcal{R}| \leq 2$ in Polynomialzeit und für $|\mathcal{R}| \geq 3$ in Pseudopolynomialzeit gelöst werden kann.

Die Arbeit von Okubo et al. (2015) betrachtet ein Maschinenbelegungsproblem, das als Mehr-Modus-Projektplanungsproblem mit reihenfolgeabhängigen Rüstzeiten und partiell erneuerbaren Ressourcen formuliert wird. Die Rüstzeiten werden durch ressourcenbeanspruchende Vorgänge modelliert, wobei die partiell erneuerbaren Ressourcen die Einhaltung der Restriktionen für den Energieverbrauch der Bearbeitungsoperationen und Rüstvorgänge sicherstellen. Für die Problemstellung wird die Zielsetzung der Projektdauerminimierung angenommen. Um die Problemstellung optimal zu lösen, werden ein ganzzahliges lineares Programm sowie ein Constraint Programming Modell formuliert. Zur Bestimmung von Näherungslösungen wird weiterhin ein Algorithmus zur Beschränkung der zu berücksichtigenden Moduszuweisungen für die verwendeten Solver vorgestellt, der die Generierungsparameter der Instanzen berücksichtigt.

In Androutsopoulos et al. (2020) wird die Slotzuteilung für Start- und Landezeiten an Flughäfen behandelt. Für die betrachtete Problemstellung werden partiell erneuerbare Ressourcen für die Modellierung der Aufnahmekapazitäten des Start- und Landebahnsystems verwendet. Die Zielsetzung setzt sich aus zwei zu minimierenden Kriterien zusammen. Das erste Zielkriterium entspricht einer mit einem konstanten Faktor multiplizierten Variante einer Earliness-Tardiness-Zielfunktion (vgl. z. B. Zimmermann et al., 2010, Abschnitt 2.1.1), in der die Differenz zwischen einem angefragten Zeitslot für den Start oder die Landung einer Fluglinie und dem zugewiesenen Zeitslot betrachtet wird. Das zweite Zielkriterium betrachtet den quadrierten Wert des ersten Zielkriteriums ohne den multiplikativen Faktor, wodurch zu starke Abweichungen der zugewiesenen Zeitslots von den angefragten Zeitslots der Fluglinien vermieden werden sollen. Für die Modellierung der Problemstellung wird jede Anfrage einer Fluglinie durch einen Vorgang mit einer Ausführungsdauer von einer Zeiteinheit dargestellt. Die Kapazitäten für die Anzahl der Starts und Landungen über bestimmmte Zeitspannen des Planungshorizonts werden durch partiell erneuerbare Ressourcen modelliert, die jeweils über zusammenhängende Zeitperioden definiert sind. Für die beschriebene Problemstellung werden Verfahren zur exakten und näherungsweisen Bestimmung aller pareto-optimalen Lösungen entwickelt, die auf der iterativen Lösung linearer Programme mit einem einzelnen Zielkriterium basieren.

In den Arbeiten von Böttcher et al. (1999) und Schirmer (1999) wurden die partiell erneuerbaren Ressourcen erstmalig im Rahmen eines Projektplanungsproblems untersucht. Beide Arbeiten behandeln Lösungsverfahren für das RCPSP/π. In Böttcher et al. (1999) wird ein exaktes Verfahren für das RCPSP/π vorgestellt, das auf dem Branch-and-Bound-Verfahren in Talbot und Patterson (1978) basiert. Zur Verbesserung der Performance des allgemeinen Enumerationsschemas, das die Vorgänge des Projekts seriell einplant, entwickeln Böttcher et al. zwei problemspezifische Zulässigkeitsschranken. Beide Schrankenwerte basieren auf der Bestimmung von Mindestressourceninanspruchnahmen der bislang noch nicht eingeplanten Vorgänge, um Teilschedules, die zu keinem zulässigen Schedule erweitert werden können, frühzeitig auszuschließen. Eine ausführliche Beschreibung der Implementierung des Verfahrens kann Böttcher (1995) entnommen werden. Weiterhin wird in Böttcher et al. (1999) ein serielles Generierungsschema zur Bestimmung von Näherungslösungen untersucht. Für das Verfahren werden verschiedene Prioritätsregeln für die zufallsbasierte Bestimmung des nächsten einzuplanenden Vorgangs sowie seines Startzeitpunkts betrachtet.

Weitere Näherungsverfahren zum RCPSP/π werden in Schirmer (1999) behandelt. Wie bereits in Böttcher et al. (1999) wird auch in Schirmer (1999) ein serielles Generierungs-

schema für das RCPSP/π untersucht. Dazu werden verschiedene Varianten des seriellen Generierungsschemas vorgestellt. Die erste Variante betrachtet einen deterministischen Ansatz zur Auswahl der einzuplanenden Vorgänge und ihrer Startzeitpunkte, wohingegen die zweite Variante eine zufallsbasierte Auswahl vornimmt. Eine weitere Variante des seriellen Generierungsschemas wählt die Vorgänge und Startzeitpunkte in Abhängigkeit von den Generierungsparametern der jeweiligen Instanz. Weiterhin werden Näherungsverfahren basierend auf dem seriellen Generierungsschema behandelt, die durch ein iteratives Vorgehen die Parametereinstellungen des Lösungsverfahrens in Abhängigkeit der Güte der Ergebnisse eigenständig anpassen. Neben den Verfahren zur Bestimmung zulässiger Schedules für das RCPSP/π werden verschiedene Verfahren zur Verbesserung gegebener Lösungen betrachtet. Zunächst werden drei verschiedene Verbesserungsverfahren untersucht, die auf einer Anpassung der Startzeitpunkte der Vorgänge eines gegebenen Schedules basieren. Darauffolgend wird eine Tabu-Suche vorgestellt, für die jede Lösung durch einen sogenannten „shift vector“ nach Sampson und Weiss (1993) repräsentiert wird. Neben den Lösungs- und Verbesserungsverfahren werden in Schirmer (1999) weiterhin verschiedene Möglickeiten der Modellierung realer Anwendungen durch partiell erneuerbare Ressourcen besprochen, die auch in Schirmer und Drexl (2001) thematisiert werden.

In Alvarez-Valdes et al. (2008) wird ein GRASP-Algorithmus zum RCPSP/π vorgestellt, der auf dem seriellen Generierungsschema aus Schirmer (1999) basiert. Für die Verbesserungsphase des GRASP-Algorithmus entwickeln die Autoren zwei verschiedene Verfahren, die durch eine Anpassung der Startzeitpunkte der Vorgänge versuchen, die Projektdauer eines gegebenen Schedules zu verbessern. Weiterhin wird der GRASP-Algorithmus in ein Preprocessing-Verfahren integriert, in dem unter anderem unzulässige Startzeitpunkte der Vorgänge eliminiert werden. Es kann gezeigt werden, dass das Preprocessing-Verfahren für einen großen Anteil der untersuchten Instanzen in der Lage ist, optimale Lösungen zu bestimmen und zu verifizieren. Weiterhin wird der GRASP-Algorithmus in verschiedenen Ausführungsvarianten untersucht, wobei sich insbesondere die Kombination mit einem Path-Relinking-Ansatz als vorteilhaft herausstellt.

In Alvarez-Valdes et al. (2006) wird ein Scatter-Search-Verfahren für das RCPSP/π entwickelt, das den GRASP-Algorithmus in Alvarez-Valdes et al. (2008) für die Generierung zulässiger Schedules verwendet. In einer experimentellen Performance-Analyse wird gezeigt, dass das Scatter-Search-Verfahren in bestimmten Ausführungsvarianten bessere Ergebnisse erzielt als der GRASP-Algorithmus in Alvarez-Valdes et al. (2008). Zugleich ist jedoch auch eine sehr viel längere Rechenzeit des Scatter-Search-Verfahrens zu beobachten. Die Zunahme der Rechenzeit wird insbesondere durch die Untersuchung des

Scatter-Search-Verfahrens in Alvarez-Valdes et al. (2015) deutlich, in der die Performance-Analyse um ein weiteres Testset mit größeren Instanzen erweitert wird.

In der Arbeit Watermeyer und Zimmermann (2018), die im Rahmen der Promotion des Autors der vorliegenden Dissertation entstanden ist, wurde das RCPSP/π erstmalig um allgemeine Zeitbeziehungen zwischen den Vorgängen des Projekts erweitert. In Watermeyer und Zimmermann (2018) wurde für das RCPSP/max-π ein Branch-and-Bound-Verfahren entwickelt, das in Watermeyer und Zimmermann (2020) im Detail ausgearbeitet und um zusätzliche Verbesserungstechniken ergänzt wurde. Das Branch-and-Bound-Verfahren sowie die verschiedenen Verbesserungstechniken werden in den nachfolgenden Kapiteln 4 und 5 näher untersucht.

Kapitel 3

Modellierungsspektrum partiell erneuerbarer Ressourcen

Die Verallgemeinerung der erneuerbaren und nicht-erneuerbaren Ressourcen durch das Konzept der partiell erneuerbaren Ressourcen wurde bereits in Böttcher et al. (1999) sowie in Schirmer (1999, Abschnitt 9.1.2) beschrieben. In diesem Kapitel wird gezeigt, dass die partiell erneuerbaren Ressourcen nicht nur die Modellierung von erneuerbaren und nicht-erneuerbaren Ressourcen ermöglichen, sondern darüber hinaus verschiedene Erweiterungskonzepte zum RCPSP umfassen. Um die unterschiedlichen Konzepte durch partiell erneuerbare Ressourcen modellieren zu können, wird in den nachfolgenden Abschnitten vorausgesetzt, dass eine maximale Projektdauer vorgegeben ist, jeder Vorgang nur zu einem ganzzahligen Zeitpunkt starten kann und alle Problemparameter ganzzahlig sind.

In Abschnitt 3.1 werden zunächst die Modellierungen von erneuerbaren und nicht-erneuerbaren Ressourcen durch partiell erneuerbare Ressourcen basierend auf den Arbeiten Böttcher et al. (1999) und Schirmer (1999, Abschnitt 9.1.2) beschrieben. Darauffolgend wird gezeigt, dass Projektplanungsprobleme mit kumulativen Ressourcen (Lagerressourcen) Spezialfälle von Projektplanungsproblemen mit partiell erneuerbaren Ressourcen und allgemeinen Zeitbeziehungen darstellen. Abschnitt 3.2 untersucht die Modellierung allgemeiner Zeitbeziehungen zwischen den Vorgängen eines Projekts und Abschnitt 3.3 die Modellierung von Kalendern durch partiell erneuerbare Ressourcen. Abschließend wird in Abschnitt 3.4 gezeigt, wie Projektdauerminimierungsprobleme mit den in diesem Kapitel betrachteten Erweiterungskonzepten durch exakte Lösungsverfahren für das RCPSP/max-π und für das RCPSP/π zumindest aus theoretischer Sicht optimal gelöst werden können.

3.1 Ressourcenkonzepte

In Böttcher et al. (1999) und Schirmer (1999, Abschnitt 9.1.2) konnten die Autoren bereits zeigen, dass sich erneuerbare und nicht-erneuerbare Ressourcen unter der Annahme ganzzahliger Startzeitpunkte der Vorgänge eines Projekts und einer vorgegebenen maximalen Projektdauer durch partiell erneuerbare Ressourcen modellieren lassen. Die Ergebnisse der beiden Arbeiten bezüglich dieses Zusammenhangs werden im Folgenden dargestellt. Jede erneuerbare Ressource $k \in \mathcal{R}^\rho$ kann durch $\bar{d}$ partiell erneuerbare Ressourcen mit $\Pi_v = \{v\}$ und $R_v = R_k^\rho$ für jede Zeitperiode $v \in \mathcal{P} := \{1, 2, \ldots, \bar{d}\}$ des Planungshorizonts modelliert werden, wobei $R_k^\rho \in \mathbb{Z}_{>0}$ der Kapazität der erneuerbaren Ressource k entspricht. Der zugehörige Ressourcenbedarf r_{iv}^d jedes Vorgangs $i \in V$ nach Ressource v wird der Inanspruchnahme $r_{ik}^\rho \in \mathbb{Z}_{\geq 0}$ der erneuerbaren Ressource k durch Vorgang i gleichgesetzt. Auf die gleiche Art und Weise können, wie in Schirmer (1999, Abschnitt 9.1.2) gezeigt wird, auch erneuerbare Ressourcen mit zeitlich variierenden Kapazitäten modelliert werden. Für jede nicht-erneuerbare Ressource $k \in \mathcal{R}^\sigma$ reicht es dagegen aus, eine einzelne partiell erneuerbare Ressource s mit $\Pi_s = \mathcal{P}$ und $R_s = R_k^\sigma$ einzuführen, wobei $R_k^\sigma \in \mathbb{Z}_{\geq 0}$ der Kapazität der nicht-erneuerbaren Ressource k entspricht. Zur Modellierung der Inanspruchnahme $r_{ik}^\sigma \in \mathbb{Z}_{\geq 0}$ der nicht-erneuerbaren Ressource k durch Vorgang i wird $r_{is}^d = r_{ik}^\sigma / p_i$ gesetzt, wobei jedes Ereignis $i \in V^e$ durch einen realen Vorgang mit $p_i = 1$ ersetzt wird. Es ist dabei zu beachten, dass $r_{ik}^\sigma / p_i \in \mathbb{Z}_{\geq 0}$ o. B. d. A. angenommen werden kann, da R_k^σ und r_{ik}^σ über alle Vorgänge $i \in V$ mit einem beliebigen positiven ganzzahligen Wert $a \in \mathbb{Z}_{>0}$ multipliziert werden können, ohne Einfluss auf den ressourcenzulässigen Bereich zu nehmen.

Im Folgenden wird gezeigt, dass kumulative Ressourcen, die auch als Lagerressourcen bezeichnet werden, unter der Annahme ganzzahliger Startzeitpunkte der Vorgänge sowie einer vorgegebenen maximalen Projektdauer durch partiell erneuerbare Ressourcen, allgemeine Zeitbeziehungen und zusätzliche Vorgänge modelliert werden können. Die kumulativen Ressourcen wurden erstmals in Schwindt (1998b) eingeführt und werden beispielsweise in den Arbeiten Neumann und Schwindt (2003), Laborie (2003) und Neumann et al. (2005) weitergehend untersucht. Die nachfolgende Notation zu den kumulativen Ressourcen ist der Arbeit Schwindt (2005) entnommen. Eine kumulative Ressource $k \in \mathcal{R}^\gamma$ kann als Lagerressource mit einer Kapazität $\overline{R}_k \in \mathbb{Z} \cup \{\infty\}$ und einem Sicherheitsbestand $\underline{R}_k \in \mathbb{Z} \cup \{-\infty\}$ interpretiert werden, die über den Zeitverlauf verbraucht und wieder aufgefüllt wird ($\underline{R}_k \leq \overline{R}_k$). In der Literatur werden diskrete und kontinuierliche kumulative Ressourcen voneinander unterschieden. Für eine diskrete kumulative Ressource $k \in \mathcal{R}^\gamma$ wird angenommen, dass der Verbrauch $r_{ik} < 0$ und die Wiederauffüllung $r_{ik} > 0$ ($r_{ik} \in \mathbb{Z}$) einer Ressource k zum Zeitpunkt des Eintritts eines Ereignisses

$i \in V^e$, wie z. B. einem Meilenstein oder dem Start- oder Endereignis eines Vorgangs, erfolgt. Das allgemeinere Konzept der kontinuierlichen kumulativen Ressourcen betrachtet dagegen auch reale Vorgänge $i \in V^r$ mit einer konstanten Verbrauchs- oder Wiederauffüllungsrate $\dot{r}_{ik} = r_{ik}/p_i$ während der Dauer ihrer Ausführungen. Wie in Neumann et al. (2003, Abschnitt 2.12.2) beschrieben wird, kann der Anteil der Ausführung eines Vorgangs $i \in V$ zu einem Zeitpunkt t für einen gegebenen Schedule S im Verhältnis zur Gesamtausführungsdauer p_i durch

$$x_i(S,t) := \begin{cases} 0, & \text{falls } t < S_i \\ 1, & \text{falls } t \geq S_i + p_i \\ (t - S_i)/p_i, & \text{sonst} \end{cases}$$

angegeben werden. Durch den Ressourcenbestand

$$r_k(S,t) := \sum_{i \in V} r_{ik}\, x_i(S,t)$$

zu einem Zeitpunkt t für einen gegebenen Schedule S können schließlich die Ressourcenrestriktionen für eine kontinuierliche kumulative Ressource $k \in \mathcal{R}^\gamma$ durch

$$\underline{R}_k \leq r_k(S,t) \leq \overline{R}_k \qquad (t \in [0, \bar{d}]) \tag{3.1}$$

beschrieben werden. Die stückweise lineare, rechtsseitig stetige Funktion $r_k(S,\cdot)$ wird im weiteren Verlauf als Ressourcenprofil bezeichnet.

Wie in der Anmerkung 1.21a in Schwindt (2005, Abschnitt 1.3.1) beschrieben wird, kann o. B. d. A. für jede Ressource $k \in \mathcal{R}^\gamma$ ein Sicherheitsbestand $\underline{R}_k = 0$ sowie eine unbegrenzte Kapazität angenommen werden ($\overline{R}_k = \infty$). Weiterhin kann $\dot{r}_{ik} = r_{ik}/p_i \in \mathbb{Z}$ ebenso o. B. d. A. vorausgesetzt werden, da die Kapazitätsgrenzen $\underline{R}_k$ und $\overline{R}_k$ sowie die Ressourceninanspruchnahmen r_{ik} aller Vorgänge $i \in V$ mit einem beliebigen positiven ganzzahligen Wert $a \in \mathbb{Z}_{>0}$ multipliziert werden können, ohne dass der ressourcenzulässige Bereich verändert wird (s. Restriktionen (3.1)). Da die Modellierung der kontinuierlichen kumulativen Ressourcen durch partiell erneuerbare Ressourcen zusätzlich die Anpassung bestehender Zeitbeziehungen zwischen den Vorgängen eines Projekts erfordert, werden die nachfolgenden Erläuterungen anhand eines Projektnetzplans verdeutlicht.

Um die einzelnen Schritte für die Transformation einer Instanz mit kontinuierlichen kumulativen Ressourcen in eine Instanz mit partiell erneuerbaren Ressourcen zu veranschaulichen, wird im Folgenden der Projektnetzplan in Abbildung 3.1 betrachtet. Das Beispiel zeigt ein Projekt mit einer kontinuierlichen kumulativen Ressource $k \in \mathcal{R}^\gamma$ mit

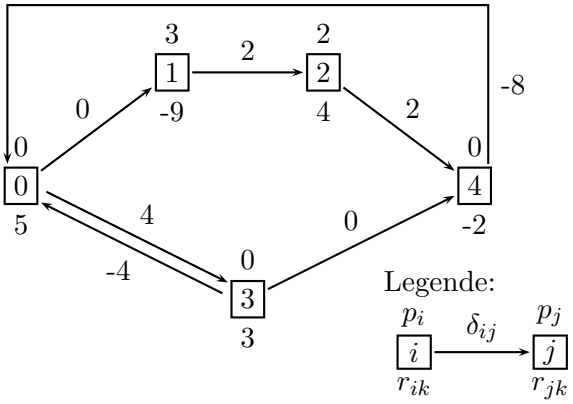

Abbildung 3.1: Beispielprojekt mit einer kontinuierlichen kumulativen Ressource

$\underline{R}_k = 0$ und $\overline{R}_k = \infty$. Die Ressource k wird durch die Ereignisse $V^e = \{0, 3, 4\}$ und die realen Vorgänge $V^r = \{1, 2\}$ des Projekts mit $\dot{r}_{1k} = -3$ und $\dot{r}_{2k} = 2$ verbraucht und erzeugt (wieder aufgefüllt).

Im ersten Transformationsschritt werden zunächst alle Ereignisse des Projekts durch reale Vorgänge ersetzt, sodass die Probleminstanz nur noch Vorgänge beinhaltet, die die kontinuierlichen kumulativen Ressourcen während ihrer Ausführungen mit einer konstanten Rate verbrauchen oder erzeugen. Hierzu wird jedes Ereignis $i \in V^e$ durch einen realen Vorgang i' mit einer Ausführungsdauer von einer Zeiteinheit ersetzt. Dadurch wird sichergestellt, dass auch die Erzeugung bzw. der Verbrauch einer Ressource durch ein Ereignis des Projekts durch partiell erneuerbare Ressourcen modelliert werden kann, da nach Definition nur reale Vorgänge eine partiell erneuerbare Ressource in Anspruch nehmen können. Als Nächstes wird jeder Zeitpunkt $t \in \mathcal{H}$ des Planungshorizonts durch eine zusätzliche Periode modelliert, wodurch die Ereignisse der ursprünglichen Probleminstanz weiterhin einem Zeitpunkt zugeordnet werden können. Entsprechend wird die Menge aller Perioden des Planungshorizonts durch $\mathcal{P}' := \{1, 2, \ldots, 2\bar{d} + 1\}$ ersetzt, wobei jeder Zeitpunkt $t \in \mathcal{H}$ durch Periode $2t + 1$ und jede Periode $v \in \mathcal{P}$ durch Periode $2v$ ersetzt wird. Um sicherzustellen, dass jeder reale Vorgang $i \in V^r$ nur während einer Periode ausgeführt werden kann, die mit einer Periode der ursprünglichen Probleminstanz übereinstimmt (gerade Periode), wird jeder Vorgang i in eine Menge $i'_1, i'_2, \ldots, i'_{p_i}$ von Vorgängen zerlegt, die jeweils eine Ausführungsdauer von einer Zeiteinheit haben. Zwischen den Vorgängen $i'_1, i'_2, \ldots, i'_{p_i}$ werden zeitliche Mindest- und Höchstabstände eingefügt, sodass Vorgang i'_{u+1} genau zwei Zeiteinheiten nach Vorgang i'_u für alle $u \in \{1, 2, \ldots, p_i - 1\}$ startet. Im weiteren Verlauf wird die Menge aller Vorgänge der transformierten Probleminstanz, die die realen Vorgänge (Ereignisse) der ursprünglichen Probleminstanz modellieren, mit V'^r (V'^e) bezeichnet. Aufgrund der zusätzlich eingefügten Zeitperioden müssen die Zeitab-

stände δ_{ij} zwischen den Startzeitpunkten der Vorgänge der ursrprünglichen Probleminstanz angepasst werden. Zur Vereinfachung wird nachfolgend angenommen, dass i' (j') dem Vorgang i'_1 (j'_1) für alle $i \in V^r$ $(j \in V^r)$ entspricht. Sei δ_{ij} ein Zeitabstand einer beliebigen Zeitbeziehung $(i,j) \in E \setminus \{(n+1,0)\}$ der ursprünglichen Probleminstanz. Dann gilt $\delta'_{i'j'} = 2\delta_{ij} + 1$ für $(i,j) \in V^e \times V^r$, $\delta'_{i'j'} = 2\delta_{ij} - 1$ für $(i,j) \in V^r \times V^e$ und $\delta'_{i'j'} = 2\delta_{ij}$ für alle verbleibende Zeitbeziehungen. Um einen Projektstart und ein Projektende zu modellieren, werden die fiktiven Vorgänge α und ω mit $p_\alpha = p_\omega = 0$ der transformierten Probleminstanz hinzugefügt. Durch Zeitbeziehungen wird sichergestellt, dass Vorgang α (ω) zur gleichen Zeit wie Vorgang $0'$ $((n+1)')$ startet (endet), und dass die maximale Projektdauer eingehalten wird $(\delta'_{\omega\alpha} = -(2\bar{d}+1))$. Abschließend werden die Ressourceninanspruchnahmen der Vorgänge der transformierten Probleminstanz bestimmt. Dafür wird jedem Vorgang $i'_1, i'_2, \ldots, i'_{p_i}$ zunächst eine Ressourceninanspruchnahme $\dot{r}_{ik} = r_{ik}/p_i$ zugeordnet. Die Vorgänge $i' \in V'^e$ behalten dagegen die Ressourceninanspruchnahme des jeweils zugehörigen Ereignisses der ursprünglichen Probleminstanz bei. Abbildung 3.2 zeigt den Projektnetzplan der transformierten Probleminstanz für das Beispielprojekt in Abbildung 3.1.

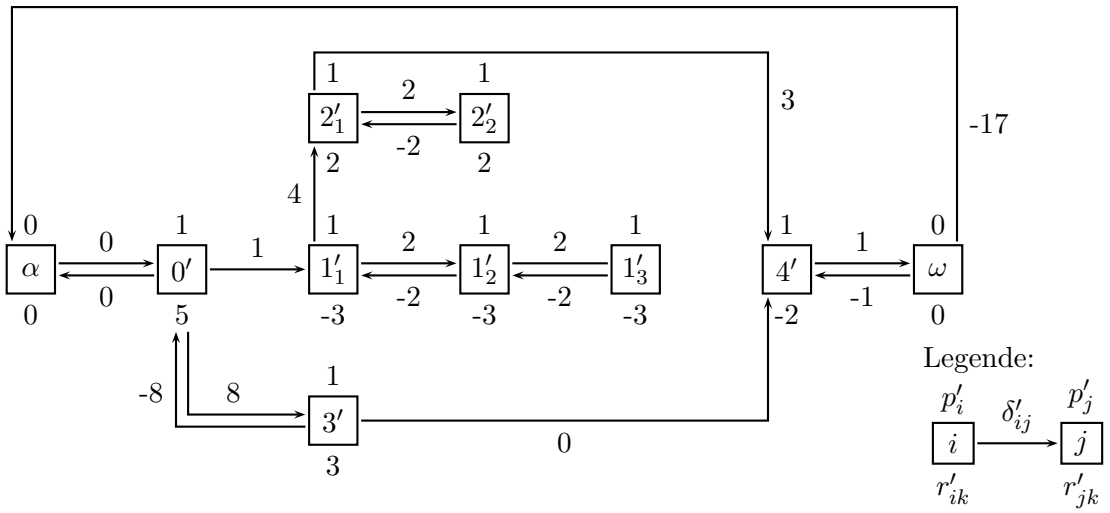

Abbildung 3.2: Projektnetzplan nach dem ersten Transformationsschritt

Im Folgenden wird gezeigt, dass ein Schedule S der ursprünglichen Probleminstanz unter der Voraussetzung $S \in \mathbb{Z}_{\geq 0}^{n+2}$ genau dann zulässig ist, wenn ein Schedule S' nach dem ersten Transformationsschritt mit $S'_{i'_u} = 2S_i + 1 + 2(u-1)$ für alle $i \in V^r$ und $u \in \{1, 2, \ldots, p_i\}$, $S'_{i'} = 2S_i$ für alle $i \in V^e$, $S'_\alpha = 2S_0$ und $S'_\omega = 2S_{n+1} + 1$ ebenfalls zulässig ist. Aus den beschriebenen Anpassungen der Zeitbeziehungen im ersten Transformationsschritt folgt zunächst, dass Schedule S genau dann zeitzulässig ist, wenn auch Schedule S' mit $S'_0 = 0$ alle Zeitrestriktionen der transformierten Probleminstanz

erfüllt. Abbildung 3.3 zeigt für einen zeitzulässigen Schedule $S = (0, 3, 5, 4, 7)$ des Beispielprojekts in Abbildung 3.1 den zugehörigen Schedule S'. Für beide Schedules sind die jeweiligen Ressourcenprofile dargestellt. Wie in Abbildung 3.3 für das Beispiel gezeigt wird, führt der erste Transformationsschritt zu einer Entfernung der Sprungstellen im Ressourcenprofil von Schedule S. Weiterhin können allgemein für jeden Zeitpunkt $t \in \mathcal{H}$ des Planungshorizonts die Zusammenhänge

$$\lim_{\tau \to t^-} r_k(S, \tau) = r_k(S, t) - \sum_{i \in \mathcal{A}^e(S,t)} r_{ik} = r'_k(S', 2t)$$

mit $\mathcal{A}^e(S, t) := \{i \in V^e \mid S_i = t\}$ und

$$\lim_{\tau \to t^+} r_k(S, \tau) = r_k(S, t) = r'_k(S', 2t + 1)$$

zwischen den Ressourcenprofilen beliebiger Schedules S und S' abgeleitet werden. Da die Bedingungen $\lim_{\tau \to t^-} r_k(S, \tau) \geq 0$ und $\lim_{\tau \to t^+} r_k(S, \tau) \geq 0$ ($r'_k(S', 2t) \geq 0$ und $r'_k(S', 2t + 1) \geq 0$) für alle $t \in \mathcal{H}$ und $k \in \mathcal{R}^\gamma$ notwendig und hinreichend für die Ressourcenzulässigkeit eines Schedules S (S') sind, folgt schließlich, dass ein Schedule S genau dann ressourcenzulässig ist, wenn Schedule S' alle Ressourcenrestriktionen der transformierten Probleminstanz erfüllt. Zusammenfassend liegt somit genau dann ein zulässiger Schedule S für die ursprüngliche Probleminstanz vor, wenn auch Schedule S' zulässig ist.

Im zweiten Transformationsschritt werden der transformierten Probleminstanz zunächst partiell erneuerbare Ressourcen hinzugefügt, die sicherstellen, dass jeder Vorgang $i' \in V'^r$ ($i' \in V'^e$) nur in einer geraden (ungeraden) Periode ausgeführt werden kann. Dazu werden zwei partiell erneuerbare Ressourcen $\Pi_1 = \{1, 3, \ldots, 2\bar{d} + 1\}$ und $\Pi_2 = \{2, 4, \ldots, 2\bar{d}\}$ mit $R_1 = R_2 = 0$ eingeführt. Die Ressourcenbedarfe der Vorgänge $i' \in V'^r$ ($i' \in V'^e$) werden durch $r^d_{i',1} = 1\,(0)$ und $r^d_{i',2} = 0\,(1)$ festgelegt. Dabei ist zu beachten, dass jeder Schedule S', der die Ressourcenrestriktionen für die partiell erneuerbaren Ressourcen Π_1 und Π_2 einhält, auch die bisherige Annahme $S \in \mathbb{Z}^{n+2}_{\geq 0}$ erfüllt. Das weitere Vorgehen basiert auf der Stetigkeit und dem konstanten Verlauf des Ressourcenprofils innerhalb jeder Zeitperiode für jeden beliebigen Schedule $S' \in \mathbb{Z}^{n+2}_{\geq 0}$ der transformierten Probleminstanz. Aufgrund dieser Eigenschaft des Ressourcenprofils ist ein Schedule S' genau dann ressourcenzulässig, wenn für alle ganzzahligen Zeitpunkte t' des Planungshorizonts $\mathcal{H}' := \{0, 1 \ldots, 2\bar{d} + 1\}$ und alle Ressourcen $k \in \mathcal{R}^\gamma$ die Bedingung $r'_k(S', t') \geq \underline{R}_k = 0$ erfüllt ist (s. Abbildung 3.3). Zu beachten ist, dass diese Eigenschaft des Ressourcenprofils eines Schedules S der ursprünglichen Probleminstanz im Allgemeinen nicht gegeben ist.

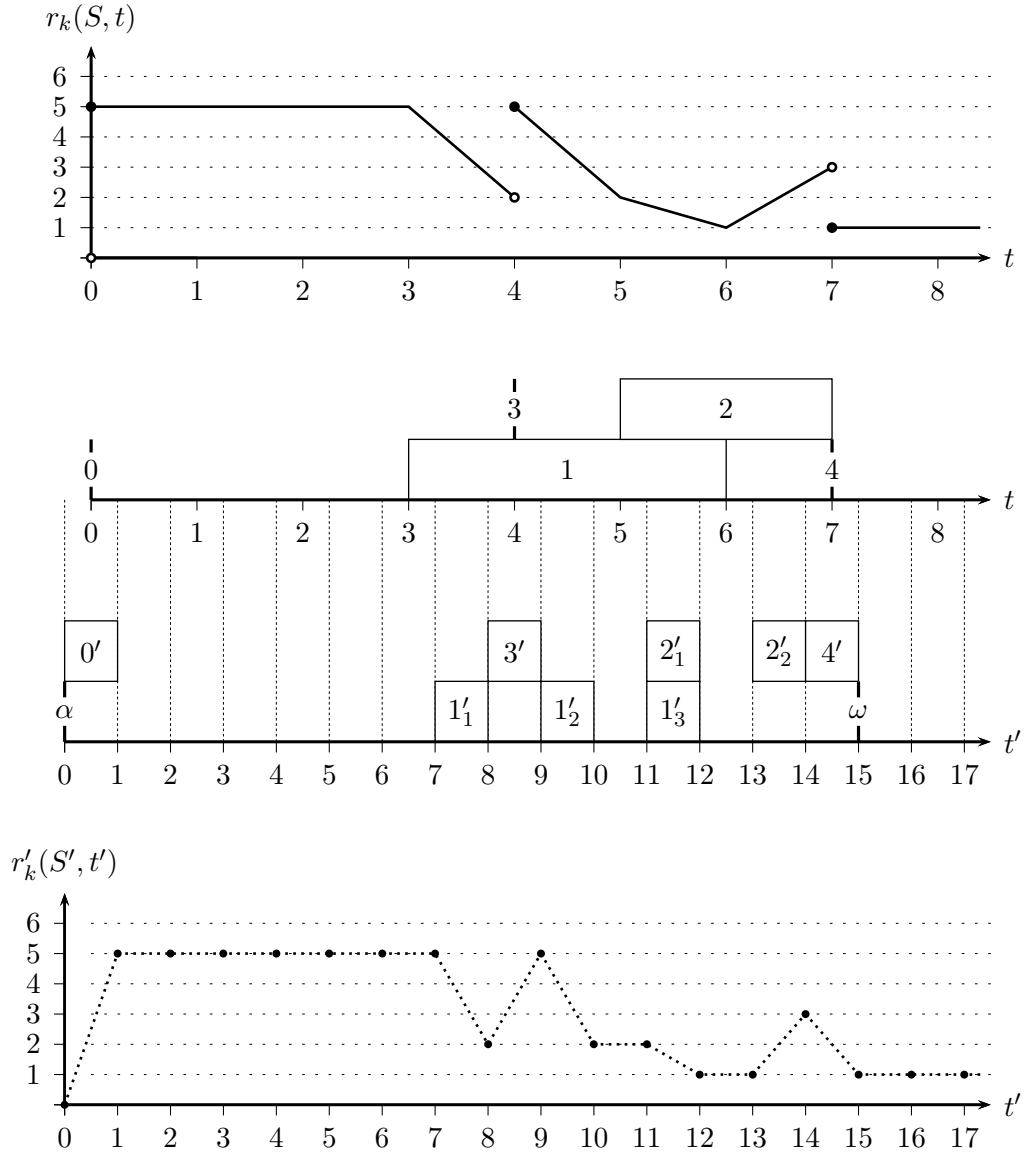

Abbildung 3.3: Gegenüberstellung der Ressourcenprofile der Schedules S und S' nach dem ersten Transformationsschritt

Im Folgenden wird gezeigt, wie die kontinuierlichen kumulativen Ressourcen der transformierten Probleminstanz durch partiell erneuerbare Ressourcen ersetzt werden können. Dazu werden weitere Bezeichner eingeführt. Zunächst gibt V' die Menge aller Vorgänge der transformierten Probleminstanz nach dem ersten Transformationsschritt an. Die Mengen der Vorgänge, die Ressource k entweder erzeugen oder verbrauchen, werden jeweils durch $V'^{+}_k := \{i \in V' \mid r'_{ik} > 0\}$ und $V'^{-}_k := \{i \in V' \mid r'_{ik} < 0\}$ definiert. Weiterhin wird $r^{c+}_k := \sum_{i \in V'^{+}_k} r'_{ik}$ verwendet, um die gesamte Erzeugungsmenge einer kontinuierlichen kumulativen Ressource k über alle Vorgänge $i \in V'$ darzustellen. Be-

trachtet sei eine beliebige Ressource $k \in \mathcal{R}^{\gamma}$ sowie ein Zeitpunkt $\tau \in \{1, 2, \ldots, 2\bar{d}\}$ des Planungshorizonts. Um die Ressourcenrestriktion $r'_k(S', \tau) \geq 0$ für den Zeitpunkt τ zu modellieren, werden drei partiell erneuerbare Ressourcen $\Pi^1_{k,\tau} = \Pi^3_{k,\tau} = \{1, 2, \ldots, \tau\}$, $\Pi^2_{k,\tau} = \{\tau + 1, \tau + 2, \ldots, 2\bar{d} + 1\}$ mit den Kapazitäten $R^1_{k,\tau} = R^2_{k,\tau} = R^3_{k,\tau} = r^{c+}_k$ hinzugefügt. Tabelle 3.1 zeigt für jeden Zeitpunkt τ die entsprechenden partiell erneuerbaren Ressourcen, die der transformierten Probleminstanz für eine kontinuierliche kumulative Ressource k hinzugefügt werden. Zur Modellierung der Ressourcenrestriktion

τ	$\Pi^1_{k,\tau}$	$\Pi^2_{k,\tau}$	$\Pi^3_{k,\tau}$
1	$\{1\}$	$\{2, 3, \ldots, 2\bar{d} + 1\}$	$\{1\}$
2	$\{1, 2\}$	$\{3, 4, \ldots, 2\bar{d} + 1\}$	$\{1, 2\}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2\bar{d}$	$\{1, 2, \ldots, 2\bar{d}\}$	$\{2\bar{d} + 1\}$	$\{1, 2, \ldots, 2\bar{d}\}$

Tabelle 3.1: Partiell erneuerbare Ressourcen zur Modellierung einer kontinuierlichen kumulativen Ressource $k \in \mathcal{R}^{\gamma}$

$r'_k(S', \tau) \geq 0$ für den Zeitpunkt τ werden weiterhin r^{c+}_k zusätzliche Vorgänge $i \in V^a_{k,\tau}$ mit Ausführungsdauern von jeweils einer Zeiteinheit ($p_i = 1$) mit Zeitabständen $\delta_{\alpha i} = 0$ und $\delta_{i\omega} = 1$ dem Projekt hinzugefügt, sodass jeder Vorgang $i \in V^a_{k,\tau}$ in jeder Periode des Planungshorizonts ausgeführt werden kann. Jedem Vorgang $i \in V^a_{k,\tau}$ wird ein Ressourcenbedarf in Höhe von einer Einheit für jede partiell erneuerbare Ressource, die für den Zeitpunkt τ hinzugefügt wurde, zugeordnet, d. h., $r^{d1}_{i,k,\tau} = r^{d2}_{i,k,\tau} = r^{d3}_{i,k,\tau} = 1$. Weiterhin werden den Vorgängen $i \in V'^+_k$ die Ressourcenbedarfe $r^{d1}_{i,k,\tau} = r^{d2}_{i,k,\tau} = r'_{ik}$ und $r^{d3}_{i,k,\tau} = 0$ zugewiesen. Für jeden ressourcenzulässigen Schedule S'' nach dem zweiten Transformationsschritt folgt, dass die Inanspruchnahme der Ressource $\Pi^2_{k,\tau}$ aller Vorgänge $i \in V'^+_k$ immer mit der Inanspruchnahme der Ressourcen $\Pi^1_{k,\tau}$ und $\Pi^3_{k,\tau}$ durch alle Vorgänge $i \in V^a_{k,\tau}$ übereinstimmt (s. Abbildung 3.4). Aus diesem Grund entspricht die verbleibende Kapazität von Ressource $\Pi^3_{k,\tau}$ der Inanspruchnahme der Ressource $\Pi^1_{k,\tau}$ durch alle Vorgänge $i \in V'^+_k$, die mit der Erzeugungsmenge der kumulativen Ressource k aller Vorgänge des Projekts über alle Zeitperioden $v \in \{1, 2, \ldots, \tau\}$ übereinstimmt. Entsprechend kann durch die Festlegung der Ressourcenbedarfe $r^{d1}_{i,k,\tau} = r^{d2}_{i,k,\tau} = 0$ und $r^{d3}_{i,k,\tau} = |r'_{ik}|$ für alle Vorgänge $i \in V'^-_k$ die gesamte Inanspruchnahme der Ressource $\Pi^3_{k,\tau}$ durch alle Vorgänge $i \in V'^-_k$ die Erzeugungsmenge der kumulativen Ressource k über alle Zeitperioden $v \in \{1, 2, \ldots, \tau\}$ nicht überschreiten.

In Abbildung 3.4 wird für den zulässigen Schedule S' aus Abbildung 3.3 die Modellierung der Ressourcenrestriktion $r'_k(S', \tau) \geq 0$ für den Zeitpunkt $\tau = 10$ dargestellt. Wie beschrieben, werden zunächst die partiell erneuerbaren Ressourcen $\Pi^1_{k,\tau} = \Pi^3_{k,\tau} =$

$\{1, 2, \ldots, 10\}$ und $\Pi^2_{k,\tau} = \{11, 12, \ldots, 17\}$ mit Kapazitäten von $r^{c+}_k = 12$ hinzugefügt. Weiterhin wird die Probleminstanz um $r^{c+}_k = 12$ zusätzliche Vorgänge $i \in V^a_{k,\tau}$ ergänzt, die in Abbildung 3.4 durch schraffierte Quadrate dargestellt werden. In der Abbildung wird ein Beispiel für die ressourcenzulässige Aufteilung der zusätzlichen Vorgänge auf die Zeitperioden in Bezug auf die partiell erneuerbaren Ressourcen dargestellt. Insbesondere geht aus der Abbildung hervor, dass die Kapazität von Ressource $\Pi^3_{k,\tau}$ nach Abzug der Inanspruchnahme durch die zusätzlichen Vorgänge in Höhe von $R^3_{k,\tau} - 4 = 8$ genau der Erzeugungsmenge der kumulativen Ressource k durch alle Vorgänge $i \in V'^+_k$ innerhalb des Zeitraums vom Projektstart bis zum Zeitpunkt $\tau = 10$ entspricht. Daraus folgt schließlich die maximal mögliche Menge an Ressourceneinheiten, die durch alle Vorgänge $i \in V'^-_k$ bis zum Zeitpunkt $\tau = 10$ verbraucht werden können. Für das Beispiel werden innerhalb der Zeitperioden $\tau' \in \{1, 2, \ldots, 10\}$ insgesamt sechs Einheiten der Ressource $\Pi^3_{k,\tau}$ durch die Vorgänge $1'_1$ und $1'_2$ in Anspruch genommen, sodass eine verbleibende Kapazität von Ressource $\Pi^3_{k,\tau}$ in Höhe von $R^3_{k,\tau} - 4 - 6 = 2$ vorliegt. Die verbleibende Kapazität stimmt dabei mit dem Ressourcenbestand $r'_k(S', \tau)$ der kumulativen Ressource k zum Zeitpunkt $\tau = 10$ überein (s. Abbildung 3.3), sodass die Einhaltung der Kapazitätsrestriktionen der partiell erneuerbaren Ressourcen sicherstellt, dass der Sicherheitsbestand $\underline{R}_k = 0$ der kumulativen Ressource k zum Zeitpunkt $\tau = 10$ nicht unterschritten wird.

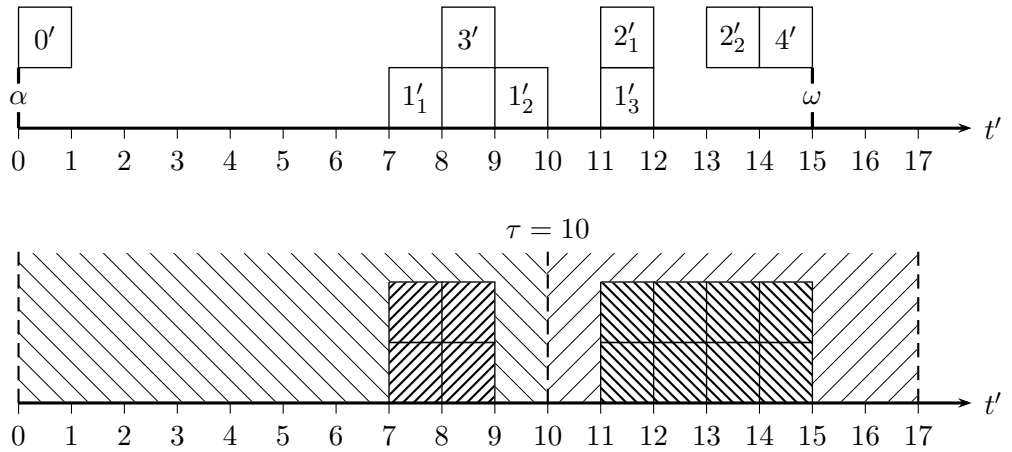

Abbildung 3.4: Modellierung der Ressourcenrestriktion einer kumulativen Ressource durch partiell erneuerbare Ressourcen für einen Zeitpunkt

Es wird nun angenommen, dass die beschriebene Transformation für alle Zeitpunkte $\tau \in \{1, 2, \ldots, 2\bar{d}\}$ und alle kumulativen Ressourcen $k \in \mathcal{R}^\gamma$ durchgeführt wurde. Dann sind die Ressourcenrestriktionen der kumulativen Ressourcen über alle Zeitpunkte $\tau \in \{1, 2, \ldots, 2\bar{d}\}$ durch einen Schedule S' genau dann erfüllt, wenn unter der Annahme $S''_i = S'_i$ für alle $i \in V'$ nach dem zweiten Transformationsschritt mindestens ein ressour-

cenzulässiger Schedule S'' existiert. Es ist weiterhin zu beachten, dass $r'_k(S', 2\bar{d}+1) \geq 0$ aus der notwendigen und hinreichenden Bedingung für die Existenz eines ressourcenzulässigen Schedules $\sum_{i \in V'} r'_{ik} \geq \underline{R}_k = 0$ für alle $k \in \mathcal{R}^\gamma$ abgeleitet werden kann und die Bedingung $r'_k(S', 0) \geq 0$ immer erfüllt ist. Es folgt schließlich unter der Voraussetzung $S''_i = S'_i$ für alle $i \in V'$, dass ein Schedule S' genau dann zulässig ist, wenn mindestens ein zulässiger Schedule S'' nach dem zweiten Transformationsschritt existiert. Zusammenfassend ergibt sich somit unter der Annahme $S''_{i''_u} = 2S_i + 1 + 2(u-1)$ für alle $i \in V^r$ und $u \in \{1, 2, \ldots, p_i\}$, $S''_{i''} = 2S_i$ für alle $i \in V^e$, $S''_\alpha = 2S_0$ und $S''_\omega = 2S_{n+1} + 1$, dass ein Schedule S der ursprünglichen Probleminstanz genau dann zulässig ist, wenn mindestens ein zulässiger Schedule S'' existiert. Entsprechend kann jeder zulässige Schedule S'' der Probleminstanz nach dem zweiten Transformationsschritt in einen zulässigen Schedule S der ursprünglichen Probleminstanz überführt werden, wobei jedem zulässigen Schedule S mindestens ein zulässiger Schedule S'' zugeordnet ist.

Abschließend ist zu beachten, dass die Untergliederung des Planungshorizonts für die Transformation in Zeitperioden, die den Zeitpunkten und Zeitperioden der ursprünglichen Probleminstanz entsprechen, nicht erforderlich ist, falls entweder nur reale Vorgänge oder nur Ereignisse mit Ressourceninanspruchnahmen betrachtet werden. Falls nur reale Vorgänge die kumulativen Ressourcen beanspruchen, können außerdem die Zerlegungen der realen Vorgänge in einzelne Ausführungseinheiten und die damit verbundenen einzufügenden zeitlichen Höchstabstände entfallen.

3.2 Allgemeine Zeitbeziehungen

In diesem Abschnitt wird gezeigt, dass unter der Annahme ganzzahliger Startzeitpunkte der Vorgänge eines Projekts sowie einer vorgegebenen maximalen Projektdauer zeitliche Mindest- und Höchstabstände zwischen den Vorgängen eines Projekts durch partiell erneuerbare Ressourcen, zusätzliche Vorgänge und Vorrangbeziehungen modelliert werden können. Im Folgenden werden die dafür erforderlichen Transformationsschritte beschrieben. Im ersten Schritt werden zunächst die fiktiven Vorgänge $i \in V^e \setminus \{0\}$ durch reale Vorgänge mit Ausführungsdauern $p'_i = 1$ ersetzt, um wie bereits bei den Ressourcenkonzepten eine Inanspruchnahme der partiell erneuerbaren Ressourcen durch die fiktiven Vorgänge zu ermöglichen. Für alle weiteren Vorgänge des Projekts gibt p'_i die Ausführungsdauer des jeweiligen Vorgangs der ursprünglichen Probleminstanz an. Als Nächstes wird der Planungshorizont um eine zusätzliche Zeitperiode $\bar{d}+1$ erweitert, sodass $\mathcal{P}' := \{1, 2, \ldots, \bar{d}+1\}$ der Menge der Zeitperioden des Planungshorizonts der transformierten Probleminstanz entspricht. Die zusätzliche Zeitperiode $\bar{d}+1$ ist dabei

erforderlich, damit jeder der fiktiven Vorgänge $i \in V^e$ mit $p'_i = 1$ auch weiterhin zum Zeitpunkt $\bar{d}$ starten kann. Im nächsten Schritt wird für jeden Vorgang $i \in V \setminus \{0\}$ eine partiell erneuerbare Ressource $\Pi_k = \{1, 2, \ldots, ES_i\} \cup \{LS_i + p'_i + 1, LS_i + p'_i + 2, \ldots, \bar{d} + 1\}$ mit $R_k = 0$ und $r^d_{ik} = 1$ eingefügt, wodurch jeder ressourcenzulässige Schedule S' der transformierten Probleminstanz die Bedingungen $ES_i \leq S'_i \leq LS_i$ für alle Vorgänge $i \in V \setminus \{0\}$ erfüllt, falls $0 \leq S'_i \leq \bar{d}$ vorausgesetzt wird. Für die Modellierung der Zeitbeziehungen wird der Probleminstanz eine weitere partiell erneuerbare Ressource $\Pi_a = \{2, 3, \ldots, \bar{d}\}$ mit $R_a = 0$ hinzugefügt. Weiterhin wird für die Modellierung einer Zeitbeziehung $(i, j) \in E$ mit $i \neq 0$ und $j \neq 0$ jeder Zeitpunkt $\tau \in \{t \in \mathcal{T}_i \mid t + \delta_{ij} > ES_j\}$ mit $\mathcal{T}_i := \{ES_i, ES_i + 1, \ldots, LS_i\}$ betrachtet. Dabei ist zu beachten, dass durch die Bedingung $\tau + \delta_{ij} > ES_j$ nur die Startzeitpunkte von Vorgang i bei der Modellierung der Zeitbeziehung berücksichtigt werden, durch die der Startzeitpunkt von Vorgang j über die zeitzulässigen Startzeitpunkte hinaus, eingeschränkt wird. Um die Zeitbeziehung (i, j) für den Startzeitpunkt τ des Vorgangs i zu modellieren, werden die partiell erneuerbaren Ressourcen $\Pi^1_\tau = \{\tau + 1, \tau + 2, \ldots, \tau + p'_i, \bar{d} + 1\}$ und $\Pi^2_\tau = \{1, 2, \ldots, \tau + \delta_{ij}\}$ mit $R^1_\tau = p'_i$ und $R^2_\tau = p'_j$ eingefügt. Den Vorgängen i und j werden die Ressourcenbedarfe $r^{d1}_{i,\tau} = r^{d2}_{j,\tau} = 1$, $r^{d2}_{i,\tau} = r^{d1}_{j,\tau} = 0$ und $r^d_{i,a} = r^d_{j,a} = 0$ zugeordnet. Abschließend wird der Probleminstanz ein weiterer Vorgang $i' \in V^a$ mit einer Ausführungsdauer von einer Zeiteinheit und den Ressourcenbedarfen $r^d_{i',a} = r^{d1}_{i',\tau} = 1$ und $r^{d2}_{i',\tau} = p'_j$ hinzugefügt. Nach der Durchführung der beschriebenen Transformationsschritte für alle Zeitbeziehungen und Startzeitpunkte der Vorgänge werden zunächst alle Zeitbeziehungen $(i, j) \in E$ entfernt. Anschließend wird ein fiktiver Vorgang ω zur Modellierung des Projektendes sowie die Zeitbeziehung $(\omega, 0)$ mit $\delta_{\omega 0} = -(\bar{d} + 1)$ zur Einhaltung der vorgegebenen maximalen Projektdauer hinzugefügt. Um sicherzustellen, dass sich kein Vorgang vor dem Projektstart 0 oder nach dem Projektende ω in Ausführung befindet, werden zusätzlich die Vorrangbeziehungen $(0, i)$ und (i, ω) mit $\delta_{0i} = 0$ und $\delta_{i\omega} = p'_i$ für alle Vorgänge $i \in (V \setminus \{0\}) \cup V^a$ ergänzt.

Abbildung 3.5 zeigt die Modellierung einer Zeitbeziehung $(i, j) \in E$ mit einem Zeitabstand $\delta_{ij} = 5$ für den Startzeitpunkt $\tau = 3$ eines realen Vorgangs i durch partiell erneuerbare Ressourcen. Für das Beispiel wird eine maximale Projektdauer $\bar{d} = 11$ angenommen. Der Vorgang i mit einer Ausführungsdauer $p'_i = 3$ nimmt genau dann die gesamte Ressourcenkapazität $R^1_\tau = p'_1 = 3$ der Ressource $\Pi^1_\tau = \{4, 5, 6, 12\}$ in Anspruch, wenn Vorgang i zum Zeitpunkt $\tau = 3$ startet. Da Π^1_τ auch die Zeitperiode $\bar{d} + 1$ beinhaltet, kann Vorgang i' durch die Ressource $\Pi_a = \{2, 3, \ldots, \bar{d}\}$ mit $R_a = 0$ nur in der ersten Periode ausgeführt werden. Entsprechend belegt Vorgang i' die gesamte Ressourcenkapazität $R^2_\tau = p'_j = 2$ der Ressource $\Pi^2_\tau = \{1, 2, \ldots, 8\}$, wodurch Vorgang j frühestens zum Zeitpunkt $\tau + \delta_{ij} = 3 + 5 = 8$ gestartet werden kann.

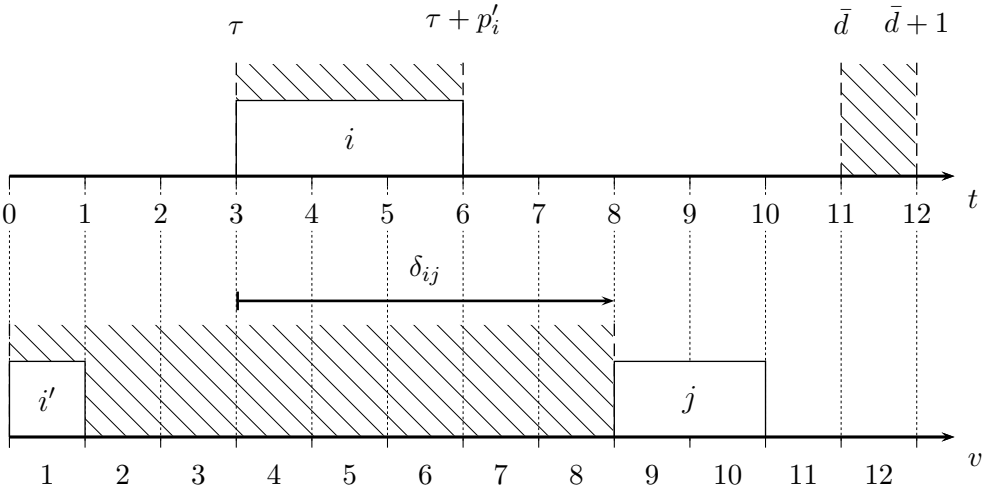

Abbildung 3.5: Modellierung allgemeiner Zeitbeziehungen durch partiell erneuerbare Ressourcen

Aus den Transformationsschritten folgt schließlich unter der Voraussetzung $S_i = S'_i$ für alle Vorgänge $i \in V$, dass ein Schedule S der ursprünglichen Probleminstanz genau dann alle Zeitrestriktionen erfüllt, wenn mindestens ein zulässiger Schedule S' der transformierten Probleminstanz existiert.

3.3 Kalendrierung

Bei der Projektplanung mit Kalendern unter der Annahme ganzzahliger Startzeitpunkte der Vorgänge wird davon ausgegangen, dass jede erneuerbare Ressource $k \in \mathcal{R}^\rho$ nur in vorgegebenen Zeitperioden des Planungshorizonts für die Ausführung der Vorgänge $i \in V$ des Projekts zur Verfügung steht. Die nachfolgende Problemstellung sowie die verwendete Notation ist dem Kapitel 2 der Arbeit Kreter (2016) entnommen. Die Bezeichner aus Kreter (2016) und deren Definitionen werden entsprechend der Annahme ganzzahliger Startzeitpunkte der Vorgänge angepasst. Insbesondere werden die Bezeichner auf Zeitperioden anstatt auf Zeitpunkten wie in Kreter (2016) definiert.

Für die Kalendrierung wird jeder erneuerbaren Ressource $k \in \mathcal{R}^\rho$ ein Kalender $b_k^{\mathcal{R}} : \mathcal{P} \mapsto \{0, 1\}$ zugeordnet. Der Ressourcenkalender weist dabei jeder Zeitperiode $v \in \mathcal{P}$, in der die Ressource k zur Verfügung steht, den Wert $b_k^{\mathcal{R}}(v) = 1$ zu, und jeder anderen Zeitperiode den Wert $b_k^{\mathcal{R}}(v) = 0$. Basierend auf den Ressourcenkalendern ergibt sich für jeden Vorgang $i \in V$ ein Vorgangskalender $b_i^V : \mathcal{P} \mapsto \{0, 1\}$, der jeder Zeitperiode $v \in \mathcal{P}$, in der alle Ressourcen, die Vorgang i beansprucht, zur Verfügung stehen, den Wert $b_i^V(v) = 1$ zuordnet, und jeder anderen Zeitperiode den Wert $b_i^V(v) = 0$. Für jeden

realen Vorgang $i \in V^r$ wird eine Anlaufphase $\epsilon_i \in \mathbb{Z}_{>0}$ angenommen, die angibt, wie viele Zeitperioden ein Vorgang i nach seinem Start S_i mindestens in Ausführung sein muss, bevor er unterbrochen werden darf. Weiterhin wird vorausgesetzt, dass jeder Vorgang i direkt nach seinem Start in jeder Zeitperiode $v \in \mathcal{P}$ mit $b_i^V(v) = 1$ ausgeführt wird, bis er eine Ausführungsdauer von p_i Zeitperioden erreicht hat. Entsprechend können durch die Anlaufphase sowohl unterbrechbare als auch nicht-unterbrechbare Vorgänge modelliert werden. Jeder Zeitbeziehung $(i,j) \in E$ wird ein Anordnungskalender $b_{ij}^E : \mathcal{P} \mapsto \{0,1\}$ mit

$$b_{ij}^E(v) := \begin{cases} \min\limits_{k \in \mathcal{R}_{ij}^E} b_k^{\mathcal{R}}(v), & \text{falls } \mathcal{R}_{ij}^E \neq \emptyset \\ 1, & \text{sonst} \end{cases}$$

und $\mathcal{R}_{ij}^E \subseteq \mathcal{R}^\rho$ zugeordnet. Der zeitliche Abstand zwischen den Startzeitpunkten der Vorgänge i und j für eine Zeitbeziehung $(i,j) \in E$ mit $\delta_{ij}^E \in \mathbb{Z}$ in Abhängigkeit vom Startzeitpunkt S_i ist durch

$$\Delta_{ij}^E(S_i) := \begin{cases} \min\left\{ t \in \mathbb{Z} \,\middle|\, \sum\limits_{\tau=S_i+1}^{t} b_{ij}^E(\tau) \geq \delta_{ij}^E \right\} - S_i, & \text{falls } \delta_{ij}^E > 0 \\ \min\left\{ t \in \mathbb{Z} \,\middle|\, \sum\limits_{\tau=t+1}^{S_i} b_{ij}^E(\tau) \leq -\delta_{ij}^E \right\} - S_i, & \text{sonst} \end{cases}$$

gegeben. Es ist zu beachten, dass die Beschränkung der Zeitbeziehungen auf Zeitabstände zwischen den Startzeitpunkten der Vorgänge keine Beschränkung der Allgemeinheit darstellt, da in Franck (1999, Abschnitt 3.1.5) gezeigt werden konnte, dass jede andere Anordnungsbeziehung durch Zeitabstände zwischen den Startzeitpunkten der Vorgänge sowie durch zusätzliche Dummy-Vorgänge darstellbar ist. Zur Bestimmung der Inanspruchnahme einer Ressource $k \in \mathcal{R}^\rho$ in jeder Zeitperiode $v \in \mathcal{P}$ für einen gegebenen Schedule S beschreibt die aktive Menge $\mathcal{A}(S,v) := \{i \in V^r \mid S_i < v \leq C_i(S_i)\}$ mit

$$C_i(S_i) := \min\left\{ t \geq S_i + p_i \,\middle|\, \sum_{\tau=S_i+1}^{t} b_i^V(\tau) = p_i \right\}$$

die Menge aller Vorgänge, die sich in Periode v in Ausführung befinden. C_i entspricht dabei dem vom Startzeitpunkt S_i abhängigen Endzeitpunkt des Vorgangs $i \in V$, sodass $\widetilde{p}_i(S_i) := C_i(S_i) - S_i$ der Ausführungsdauer von Vorgang i entspricht. Es werden zwei Arten von Ressourcen unterschieden, die entweder während der Unterbrechung eines Vorgangs für andere Vorgänge freigegeben werden ($\rho_k = 0$) oder solange belegt bleiben, bis der Vorgang beendet wird ($\rho_k = 1$). Für jede Ressource $k \in \mathcal{R}^\rho$ mit $\rho_k = 1$ wird angenommen, dass sie in jeder Zeitperiode des Planungshorizonts zur Verfügung steht, d. h., $b_k^{\mathcal{R}}(v) = 1$ für alle $v \in \mathcal{P}$. Erneuerbare Ressourcen mit $\rho_k = 1$ werden beispielsweise

zur Modellierung von Maschinen mit aufwändigen Rüstvorgängen eingesetzt. Die Inanspruchnahme einer Ressource $k \in \mathcal{R}^\rho$ für einen gegebenen Schedule S in einer Zeitperiode $v \in \mathcal{P}$ entspricht

$$r_k^{cal}(S,v) := \sum_{i \in \mathcal{A}(S,v): b_i^V(v)=1} r_{ik}^\rho + \sum_{i \in \mathcal{A}(S,v): b_i^V(v)=0} r_{ik}^\rho \, \rho_k.$$

Der zulässige Bereich der ressourcenbeschränkten Projektplanung mit allgemeinen Zeitbeziehungen und Kalendern für eine beliebige Zielfunktion wird durch die folgenden Restriktionen beschrieben:

$$r_k^{cal}(S,v) \leq R_k \qquad (k \in \mathcal{R}^\rho,\ v \in \mathcal{P}) \tag{3.2}$$

$$S_j - S_i \geq \Delta_{ij}^E(S_i) \qquad ((i,j) \in E) \tag{3.3}$$

$$\sum_{v=S_i+1}^{S_i+\epsilon_i} b_i^V(v) = \epsilon_i \qquad (i \in V^r) \tag{3.4}$$

$$S_0 = 0,\ S_{n+1} \leq \bar{d} \tag{3.5}$$

$$S_i \in \mathbb{Z}_{\geq 0} \qquad (i \in V) \tag{3.6}$$

Im Folgenden wird die Transformation einer Probleminstanz der ressourcenbeschränkten Projektplanung mit allgemeinen Zeitbeziehungen und Kalendern in eine Probleminstanz mit Vorrangbeziehungen und partiell erneuerbaren Ressourcen beschrieben. Im ersten Schritt wird zunächst jeder reale Vorgang $i \in V^r$ des Projekts in eine Menge $i'_1, i'_2, \ldots, i'_{p_i}$ von Vorgängen zerlegt, die jeweils eine Ausführungsdauer von einer Zeiteinheit haben. Die Zeitbeziehungen (3.3) zwischen den Vorgängen des Projekts werden durch partiell erneuerbare Ressourcen und zusätzliche Vorgänge modelliert, indem die Transformationsschritte aus Abschnitt 3.2 mit einigen Anpassungen durchgeführt werden. Zunächst werden die in Abschnitt 3.2 betrachteten Vorgänge $i, j \in V^r$ durch die Vorgänge i'_1, j'_1 ersetzt. Weiterhin werden statt der frühesten und spätesten zeitzulässigen Startzeitpunkte ES_i und LS_i die frühesten und spätesten Startzeitpunkte ES_i^{cal} und LS_i^{cal} der Vorgänge betrachtet, die die Restriktionen (3.3) bis (3.6) erfüllen. ES_i^{cal} und LS_i^{cal} werden als frühester und spätester kalenderzulässiger Startzeitpunkt bezeichnet und können durch Label-Correcting-Verfahren, die in Franck et al. (2001a) entwickelt wurden, bestimmt werden. Die letzte Anpassung, die in Abschnitt 3.2 vorgenommen werden muss, ist die Betrachtung des Zeitabstands $\Delta_{ij}^E(\tau)$ bzw. $\Delta_{ij}^E(t)$ an Stelle von δ_{ij}. Im nächsten Schritt werden zwischen den Vorgängen $i'_1, i'_2, \ldots, i'_{p_i}$ jedes realen Vorgangs $i \in V^r$ Vorrangbeziehungen eingefügt, sodass Vorgang i'_{u+1} frühestens nach dem Ende von Vorgang i'_u für alle $u \in \{1, 2, \ldots, p_i - 1\}$ gestartet werden kann. Um sicherzustellen, dass die Vorgänge

$i'_1, i'_2, \ldots, i'_{p_i}$ nacheinander ohne Unterbrechung in einer Periode $v \in \mathcal{P}$ mit $b_i^V(v) = 1$ ausgeführt werden, wird zusätzlich die Zeitbeziehung $S_{i'_1} - S_{i'_{p_i}} \geq -(C_i(S_{i'_1}) - 1)$ hinzugefügt und durch partiell erneuerbare Ressourcen modelliert. Zur Vereinfachung der nachfolgenden Erläuterungen bezeichnet V'^r die Menge aller Vorgänge der transformierten Probleminstanz, die jeweils einer Ausführungseinheit eines realen Vorgangs $i \in V^r$ der ursprünglichen Probleminstanz entsprechen. Damit jeder Vorgang $i' \in V'^r$ nur innerhalb einer Zeitperiode $v \in \mathcal{P}$ mit $b_i^V(v) = 1$ ausgeführt werden kann, wird für jeden Vorgang $i \in V^r$ eine partiell erneuerbare Ressource $\Pi_b = \{v \in \mathcal{P} \mid b_i^V(v) = 0\}$ mit $R_b = 0$ und $r^d_{i'_1,b} = r^d_{i'_2,b} = \ldots = r^d_{i'_{p_i},b} = 1$ definiert. Im nächsten Schritt wird die Anlaufphase ϵ_i jedes Vorgangs $i \in V^r$ berücksichtigt. Dazu wird für jeden Vorgang $i \in V^r$ eine partiell erneuerbare Ressource $\Pi_\epsilon = \{v \in \mathcal{P} \mid \sum_{v'=v}^{v+\epsilon_i-1} b_i^V(v') < \epsilon_i\}$ mit $R_\epsilon = 0$ und $r^d_{i'_1\epsilon} = 1$ hinzugefügt. Dadurch wird sichergestellt, dass der Vorgang i'_1, der die erste Ausführungseinheit von Vorgang $i \in V^r$ darstellt, nur in den Perioden, die die Anlaufphase ermöglichen, gestartet werden kann. Im letzten Transformationsschritt werden die erneuerbaren Ressourcen $k \in \mathcal{R}^\rho$, wie in Abschnitt 3.1 beschrieben, durch partiell erneuerbare Ressourcen ersetzt und den Vorgängen die entsprechenden Ressourcenbedarfe zugeordnet. Um die Restriktionen (3.2) durch partiell erneuerbare Ressourcen zu modellieren, müssen weiterhin die Inanspruchnahmen der Ressourcen $k \in \mathcal{R}^\rho$ mit $\rho_k = 1$ während der Unterbrechung der Vorgänge berücksichtigt werden. Im Folgenden wird eine Menge $Q := \{s, s+1, \ldots, e\} \subseteq \mathcal{P} \setminus \{1, \bar{d}\}$ mit $b_i^V(s) = b_i^V(s+1) = \ldots = b_i^V(e) = 0$ und $b_i^V(s-1) = b_i^V(e+1) = 1$ als Nicht-Arbeitsintervall eines Vorgangs $i \in V^r$ bezeichnet. Für jedes Nicht-Arbeitsintervall Q eines Vorgangs $i \in V^r$ (mit $\epsilon_i < p_i$) werden $|Q|$ Vorgänge $i' \in V^a$ mit einer Ausführungsdauer von jeweils einer Zeiteinheit hinzugefügt. Damit diese Vorgänge nur in den Perioden $v \in Q^+ := \{s-1, s, \ldots, e+1\}$ ausgeführt werden können, wird eine partiell erneuerbare Ressource $\Pi_1^q = \mathcal{P} \setminus Q^+$ mit $R_1^q = 0$ und $r^{dq}_{i',1} = 1$ für alle $i' \in V^a$ definiert. Weiterhin wird für jede Zeitperiode $v \in Q$ eine partiell erneuerbare Ressource $\Pi_v^q = \{v\}$ mit $R_v^q = 1$ und $r^{dq}_{i',v} = 1$ für alle Vorgänge $i' \in V^a$ eingeführt, um sicherzustellen, dass die Vorgänge $i' \in V^a$ in keiner Periode $v \in Q$ zeitgleich ausgeführt werden. Da die Vorgänge $i' \in V^a$ die Inanspruchnahmen aller Ressourcen $k \in \mathcal{R}^\rho$ mit $\rho_k = 1$ während einer Unterbrechung des Vorgangs $i \in V^r$ im Nicht-Arbeitsintervall Q modellieren, wird jedem Vorgang $i' \in V^a$ ein Bedarf $r^d_{i',v} = r^\rho_{ik}$ für jede partiell erneuerbare Ressource, die jeweils die Kapazität einer Ressource $k \in \mathcal{R}^\rho$ mit $\rho_k = 1$ in Periode $v \in Q$ modelliert, zugewiesen. Abschließend wird die partiell erneuerbare Ressource $\Pi_s^q = \{s-1, e+1\}$ mit $R_s^q = 2|Q|$ definiert. Der Bedarf nach der Ressource Π_s^q wird für jeden Vorgang $i'_1, i'_2, \ldots, i'_{p_i}$ auf $|Q|$ Einheiten und für alle Vorgänge $i' \in V^a$ auf eine Einheit gesetzt. Entsprechend stellt die letzte Ressource sicher, dass die Vorgänge $i' \in V^a$ in den Perioden der Menge Q ausgeführt werden, falls sich der Vorgang $i \in V^r$

während der Perioden $s-1$ und $e+1$ in Ausführung befindet. Aus den Transformationsschritten folgt schließlich unter der Voraussetzung $S'_{i'_1} = S_i$ für alle $i \in V^r$ und $S'_{i'} = S_i$ für alle $i \in V^e$, dass ein Schedule S der ursprünglichen Probleminstanz genau dann zulässig ist, wenn mindestens ein zulässiger Schedule S' der transformierten Probleminstanz existiert.

3.4 Implikationen

In diesem Abschnitt wird gezeigt, dass Projektdauerminimierungsprobleme, die sich aus den Erweiterungskonzepten der vorangegangenen Abschnitte ergeben, durch exakte Lösungsverfahren sowohl für das RCPSP/max-π als auch für das RCPSP/π optimal gelöst werden können. Im Folgenden werden zunächst die Zusammenhänge zwischen dem RCPSP/max-π und unterschiedlichen Projektdauerminimierungsproblemen mit allgemeinen Zeitbeziehungen untersucht. Dazu werden das RCPSP/max mit erneuerbaren Ressourcen, das RCPSP/max-c mit diskreten kumulativen Ressourcen, das RCPSP/max-cc mit kontinuierlichen kumulativen Ressourcen sowie das RCPSP/max-cal mit erneuerbaren Ressourcen und Kalendern betrachtet. Alle Parameter der Problemstellungen werden als ganzzahlig angenommen. Weiterhin wird zunächst für den ersten Teil dieses Abschnitts vorausgesetzt, dass die Startzeitpunkte aller Vorgänge ganzzahlig sind. Da für die Projektplanungsprobleme ohne Kalender mit $\bar{d} := \sum_{i \in V} \max(p_i, \max_{(i,j) \in E} \delta_{ij})$ eine obere Schranke für die kürzeste Projektdauer gegeben ist, kann jede Instanz des RCPSP/max, RCPSP/max-c oder RCPSP/max-cc mit einer vorgegebenen maximalen Projektdauer $\bar{d}$ in eine Probleminstanz des RCPSP/max-π, die alle optimalen Schedules enthält, gemäß der Beschreibungen in den vorangegangenen Abschnitten transformiert werden. Für das RCPSP/max-cal (mit $\rho_k = 1$ für alle $k \in \mathcal{R}^\rho$) konnte in Franck et al. (2001a) eine obere Schranke $\bar{d}^{cal} := \sum_{i \in V} \max(\bar{p}_i, \max_{j \in Succ(i)} \bar{\Delta}_{ij})$ für die kürzeste Projektdauer über alle zulässigen Schedules entwickelt werden. $\bar{p}_i$ und $\bar{\Delta}_{ij}$ stellen jeweils eine obere Schranke für die Ausführungsdauern der Vorgänge und für die Zeitabstände der Zeitbeziehungen dar. Eine detaillierte Erläuterung zur Berechnung von $\bar{d}^{cal}$ kann den Arbeiten Franck et al. (2001a) und Neumann et al. (2003, Abschnitt 2.11) entnommen werden. Durch $\bar{d}^{cal}$ kann schließlich auch für eine Instanz des RCPSP/max-cal mit einer vorgegebenen maximalen Projektdauer, die alle optimalen Schedules enthält, eine Probleminstanz des RCPSP/max-π, wie in Abschnitt 3.3 beschrieben, abgeleitet werden. Wie aus den vorangegangenen Abschnitten hervorgeht, ist jedem optimalen Schedule S'^* der jeweils generierten RCPSP/max-π-Instanz genau ein optimaler Schedule S^* der ursprünglichen Instanz zugeordnet und jedem Schedule S^* mindestens ein Schedule S'^*.

Da die Rangfolge der zulässigen Schedules bezüglich der Projektdauer durch die Transformationen nicht verändert wird, kann für die RCPSP/max-π-Instanz ein optimaler Schedule bestimmt und in eine optimale Lösung des jeweiligen Projektdauerminimierungsproblems überführt werden. Weiterhin kann dadurch, dass nach Abschnitt 3.2 jede allgemeine Zeitbeziehung durch partiell erneuerbare Ressourcen, zusätzliche Vorgänge und Vorrangbeziehungen modellierbar ist, jedes der Projektdauerminimierungsprobleme auch durch ein exaktes Verfahren für das RCPSP/π optimal gelöst werden.

Im Folgenden wird die Annahme ganzzahliger Startzeitpunkte der Vorgänge für die Projektdauerminimierungsprobleme RCPSP/max, RCPSP/max-c, RCPSP/max-cc und RCPSP/max-cal verworfen und stattdessen von reellwertigen Startzeitpunkten ausgegangen ($S_i \in \mathbb{R}_{\geq 0}$). Da der zulässige Bereich des RCPSP/max und des RCPSP/max-c jeweils einer Vereinigung endlich vieler ganzzahliger Polyeder entspricht (vgl. z. B. Bartusch et al., 1988; Neumann und Schwindt, 2003), existiert für beide Projektplanungsprobleme genau dann mindestens eine ganzzahlige optimale Lösung, wenn eine zulässige Lösung existiert. Entsprechend können das RCPSP/max und das RCPSP/max-c auch weiterhin unter der Annahme reellwertiger Startzeitpunkte der Vorgänge durch ein exaktes Verfahren für das RCPSP/max-π oder für das RCPSP/π optimal gelöst werden. Die gleiche Aussage gilt auch für das RCPSP/max-cal mit reellwertigen Startzeitpunkten, da in Kreter (2016, Abschnitt 5.2) gezeigt werden konnte, dass mindestens eine optimale Lösung ganzzahlig ist, falls eine zulässige Lösung existiert. Im Gegensatz zu RCPSP/max, RCPSP/max-c und RCPSP/max-cal kann für das RCPSP/max-cc lediglich die Aussage getroffen werden, dass der zulässige Bereich eine Vereinigung endlich vieler rationaler Polyeder darstellt. Basierend auf den Branch-and-Bound-Verfahren, die in Schwindt (2005, Abschnitt 5.4) und Neumann et al. (2003, Abschnitt 2.12) beschrieben werden, kann allerdings abgeleitet werden, dass es mindestens eine rationalwertige optimale Lösung gibt, die sich durch die ganzzahligen Parameter der Probleminstanz darstellen lässt, falls eine zulässige Lösung existiert. Als Folgerung daraus kann das RCPSP/max-cc mit reellwertigen Startzeitpunkten der Vorgänge durch exakte Verfahren für das RCPSP/max-π und für das RCPSP/π optimal gelöst werden, falls jede Zeitperiode des Planungshorizonts in eine ausreichende Anzahl von Teilperioden gleicher Länge zerlegt wird.

Abschließend wird erläutert, wie die NP-Schwere im strengen Sinne des RCPSP/max-π aus den Transformationsschritten in Abschnitt 3.1 zu den kumulativen Ressourcen abgeleitet werden kann. Dazu sei allerdings erwähnt, dass diese Herleitung lediglich eine Möglichkeit darstellt, die Komplexitätsklasse für das RCPSP/max-π durch eine der beschriebenen Transformationen zu bestimmen, wohingegen die strenge NP-Schwere des RCPSP/max-π auch direkt aus der strengen NP-Schwere des RCPSP/π folgt, die bereits

in Schirmer (1999, Abschnitt 9.5.2) bewiesen wurde. Analog zum Beweis von Theorem 3 in Neumann und Schwindt (2003) kann zunächst gezeigt werden, dass die Zulässigkeitsvariante des RCPSP/max-c mit ganzzahligen Startzeitpunkten der Vorgänge und mit einer vorgegebenen maximalen Projektdauer NP-vollständig im strengen Sinne ist. Weiterhin stellt die Transformation in Abschnitt 3.1 eine pseudopolynomielle Transformation der Zulässigkeitsvariante des RCPSP/max-c auf die Zulässigkeitsvariante des RCPSP/max-π dar (vgl. Garey und Johnson, 1979, Abschnitt 4.2). Da analog zum Beweis von Theorem 9.1 in Schirmer (1999) gezeigt werden kann, dass die Zulässigkeitvariante des RCPSP/max-π in der Problemklasse NP liegt, folgt schließlich die NP-Schwere im strengen Sinne für das RCPSP/max-π.

Kapitel 4

Projektplanung unter Startzeitbeschränkungen

In jedem Enumerationsknoten der Branch-and-Bound-Verfahren zum RCPSP/max-π, die nachfolgend in Kapitel 5 behandelt werden, wird ein Projektdauerminimierungsproblem mit allgemeinen Zeitbeziehungen und unter Einschränkung der Startzeitpunkte der Vorgänge gelöst. Die Beschränkung der Startzeitpunkte der Vorgänge auf eine Teilmenge aller Zeitpunkte des Planungshorizonts wird durch eine sogenannte Startzeitbeschränkung W modelliert, die dem Startzeitpunkt jedes Vorgangs $i \in V$ einen Wertebereich W_i zuordnet ($S_i \in W_i$).

Definition 4.1. *Ein Vektor $W := (W_i)_{i \in V}$ mit $W_i \subseteq \mathcal{H}$ für alle $i \in V$ und $W_0 = \{0\}$ wird Startzeitbeschränkung mit W_i als Startzeitbeschränkung von Vorgang i genannt.*

Das Projektdauerminimierungsproblem unter Startzeitbeschränkungen (PS(W)) wird durch das folgende mathematische Modell beschrieben:

$$\left.\begin{array}{lll} \text{Minimiere} & S_{n+1} & \\ \text{u. d. N.} & S_j - S_i \geq \delta_{ij} & ((i,j) \in E) \\ & S_i \in W_i & (i \in V) \end{array}\right\} \text{(PS}(W)\text{)}$$

Der zulässige Bereich $\mathcal{S}_T(W) := \{S \in \mathcal{S}_T \mid S_i \in W_i \text{ für alle } i \in V\}$ wird als Menge aller W-zulässigen Schedules bezeichnet und jeder Schedule $S \in \mathcal{S}_T(W)$ W-zulässig genannt. Weiterhin wird jeder Startzeitpunkt $t \in W_i$ eines Vorgangs $i \in V$, für den ein Schedule $S \in \mathcal{S}_T(W)$ mit $S_i = t$ existiert, als W-zulässig bezeichnet. Wie in Kapitel 5 gezeigt wird, stellt der zulässige Bereich des Problems (PS(W)) den Suchraum eines Enumerationsknotens in jedem der in dieser Arbeit entwickelten Branch-and-Bound-Verfahren dar, sodass $\mathcal{S}(W) := \mathcal{S}_T(W) \cap \mathcal{S}$ dem zulässigen Bereich eines Enumerationsknotens entspricht.

In Abschnitt 4.1 werden zunächst verschiedene Zeitplanungsverfahren für das Projektdauerminimierungsproblem unter Startzeitbeschränkungen vorgestellt. Basierend auf den Zeitplanungsverfahren werden in Abschnitt 4.2 Konsistenztests entwickelt, die die Startzeitpunkte der Vorgänge des Projekts bei einer gegebenen Startzeitbeschränkung weiter reduzieren können. Abschließend werden in Abschnitt 4.3 untere Schranken für die kürzeste Projektdauer über alle Schedules im zulässigen Bereich des Problems (PS(W)) erläutert.

Die Inhalte dieses Kapitels wurden bereits weitestgehend in der Arbeit Watermeyer und Zimmermann (2020) behandelt, die im Rahmen der Promotion des Autors der vorliegenden Dissertation entstanden ist.

4.1 Zeitplanungsverfahren

In diesem Abschnitt werden Zeitplanungsverfahren für die Projektplanung unter Startzeitbeschränkungen vorgestellt. In Abschnitt 4.1.1 werden zunächst Label-Correcting-Verfahren zur Bestimmung der frühesten und spätesten Startzeitpunkte der Vorgänge besprochen. Anschließend wird in Abschnitt 4.1.2 auf die Bestimmung indirekter Zeitabstände zwischen allen Vorgangspaaren des Projekts eingegangen, die aus den Zeitplanungsverfahren in Abschnitt 4.1.1 hergeleitet werden. Weiterhin wird gezeigt, dass die Zeitplanungsverfahren in Abschnitt 4.1.2 alle W-zulässigen Startzeitpunkte der Vorgänge eines Projekts bestimmen.

4.1.1 Früheste und späteste Startzeitpunkte

Das erste Zeitplanungsverfahren bestimmt die frühesten zeitzulässigen Startzeitpunkte aller Vorgänge des Projekts unter Berücksichtigung einer vorgegebenen Startzeitbeschränkung W und einem vorgegebenen frühesten Startzeitpunkt t_α eines Vorgangs $\alpha \in V$. Das Verfahren entspricht einem Label-Correcting-Algorithmus, der entweder den eindeutigen Minimalpunkt der Menge $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha) := \{S \in \mathcal{S}_T(W) \mid S_\alpha \geq t_\alpha\}$ bestimmt oder zeigt, dass kein Schedule $S \in \widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha)$ existiert. Im Folgenden bezeichnet $ES(W, \alpha, t_\alpha)$ den eindeutigen Minimalpunkt der Menge $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha)$, wobei $ES_i(W, \alpha, t_\alpha)$ dem frühestmöglichen W-zulässigen Startzeitpunkt von Vorgang $i \in V$, unter der Voraussetzung, dass Vorgang $\alpha \in V$ nicht früher als zum Zeitpunkt t_α startet, entspricht. Da $\widetilde{S}_T(W, 0, 0) = \mathcal{S}_T(W)$ gilt, kann das Projektdauerminimierungsproblem unter Startzeitbeschränkungen (PS(W)) durch das erste Zeitplanungsverfahren mit $\alpha := 0$ und $t_\alpha := 0$

optimal gelöst werden. Im Folgenden wird der eindeutige Minimalpunkt des zulässigen Bereichs von (PS(W)) durch den frühesten W-zulässigen Schedule $ES(W) := \min \mathcal{S}_T(W)$ angegeben. Die Betrachtung der unteren Schranke t_α für den Startzeitpunkt S_α eines beliebigen Vorgangs $\alpha \in V$ ermöglicht, wie in Abschnitt 4.1.2 gezeigt wird, die Entwicklung eines Verfahrens zur Bestimmung der indirekten Zeitabstände zwischen allen Vorgangspaaren des Projekts.

Algorithmus 4.1 beschreibt das erste Zeitplanungsverfahren zur Bestimmung des eindeutigen Minimalpunkts von $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha) \neq \emptyset$. Für das Verfahren wird jedem Vorgang $i \in V$ eine Markierung ν_i zugeordnet, die in jeder Iteration als vorübergehender frühester W-zulässiger Startzeitpunkt unter der Bedingung $S_\alpha \geq t_\alpha$ interpretiert werden kann. Zur Vereinfachung der Darstellung wird in Algorithmus 4.1 der Zeitabstand zwischen dem frühesten Startzeitpunkt $\nu_j \in W_j$ mit $\nu_j \geq \nu_i + \delta_{ij}$ eines Vorgangs $j \in V$ und dem Startzeitpunkt ν_i eines Vorgangs $i \in V$ für jede Zeitbeziehung $(i, j) \in E$ durch

$$\widetilde{\delta}_{ij}(W, \nu_i) := \begin{cases} \min\{\tau \in W_j \mid \tau \geq \nu_i + \delta_{ij}\} - \nu_i, & \text{falls } \{\tau \in W_j \mid \tau \geq \nu_i + \delta_{ij}\} \neq \emptyset \\ \delta_{ij}, & \text{sonst} \end{cases}$$

beschrieben (falls $\nu_i + \delta_{ij} \leq \max W_j$). Zur Veranschaulichung der Definition von $\widetilde{\delta}_{ij}(W, \nu_i)$ zeigt Abbildung 4.1 ein Projektplanungsproblem mit Startzeitbeschränkungen $W_0 = \{0\}$, $W_1 = \{6, 7, 8\}$, $W_2 = \{2, 3, 9, 10\}$ und $W_3 = \mathcal{H}$, wobei W_1 und W_2 durch schraffierte Flächen angedeutet werden. Für einen W-zulässigen Schedule $S = (0, 7, 9, 12)$ werden die Zeitabstände δ_{ij} und $\widetilde{\delta}_{ij}(W, S_i)$ zwischen dem Projektstart und den realen Vorgängen einander gegenübergestellt. Beispielsweise entspricht $S_1 + \widetilde{\delta}_{12}(W, S_1) = 7 + 2 = 9$ dem frühesten Startzeitpunkt von Vorgang 2 mit $S_2 \geq S_1 + \delta_{12} = 7 - 2 = 5$ und $S_2 \in W_2$.

Im ersten Schritt von Algorithmus 4.1 werden die Bedingungen $\mathcal{S}_T = \emptyset$, $\exists i \in V : W_i = \emptyset$ und $t_\alpha > \max W_\alpha$ überprüft. Aus der Definition von $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha)$ geht direkt hervor, dass jede dieser Bedingungen, falls sie erfüllt ist, $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha) = \emptyset$ impliziert. Falls keine der Bedingungen erfüllt ist, wird Vorgang α mit dem Zeitpunkt $t'_\alpha := \min\{\tau \in W_\alpha \mid \tau \geq t_\alpha\}$ markiert, allen weiteren Vorgängen $i \in V \setminus \{\alpha\}$ eine Markierung $\nu_i := -\infty$ zugewiesen und die Schlange $Q := \{\alpha\}$ initialisiert. In jeder Iteration des Algorithmus 4.1 werden die Markierungen aller direkten Nachfolger $j \in Succ(i)$ aller Vorgänge $i \in V$ im Projektnetzplan N, die in der Initialisierung oder der direkt vorangegangenen Iteration der Schlange Q hinzugefügt wurden, überprüft. Falls $\nu_i + \widetilde{\delta}_{ij}(W, \nu_i) > \nu_j$ gilt, wird ν_j durch $\nu_j := \nu_i + \widetilde{\delta}_{ij}(W, \nu_i)$ aktualisiert. Da durch die Erhöhung der Markierung von Knoten j alle direkten Nachfolger von j im Projektnetzplan N in der nächsten Iteration überprüft werden müssen, wird Vorgang j der Schlange Q hinzugefügt, falls j noch nicht in Q vor-

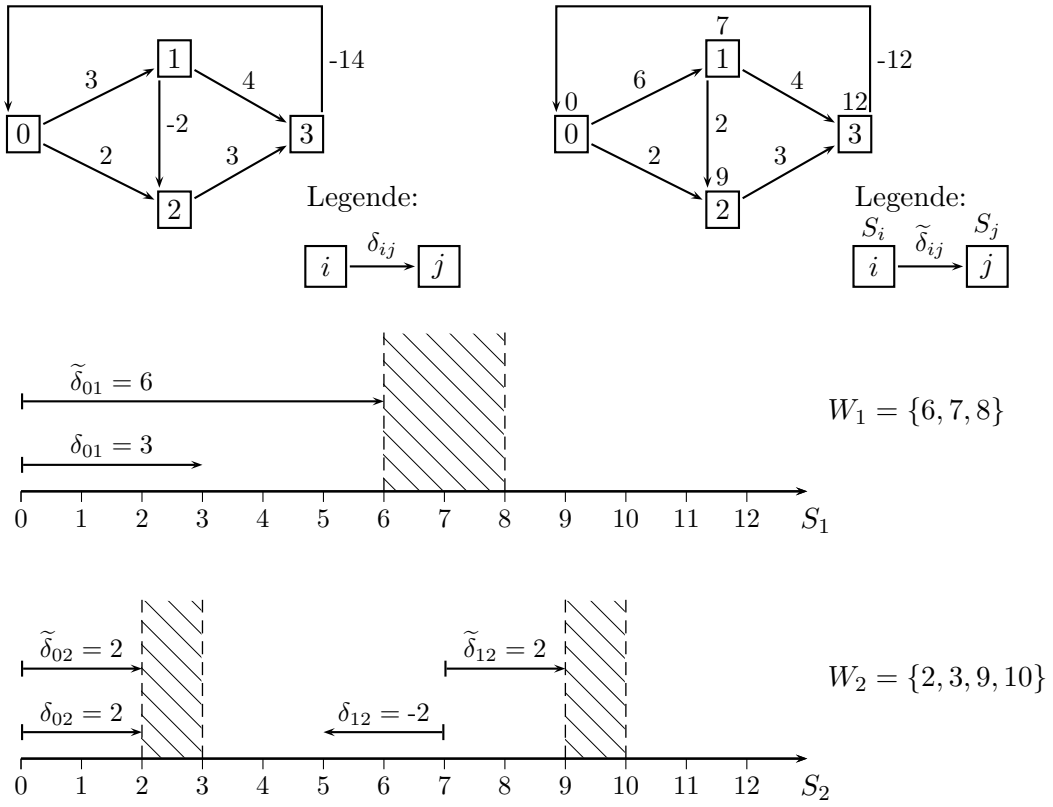

Abbildung 4.1: Zeitabstände bei Startzeitbeschränkungen

handen ist. Im Fall, dass nach der Aktualisierung die Markierung ν_j größer als $\max W_j$ ist, terminiert das Verfahren, da $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha) = \emptyset$ gilt (s. Beweis von Satz 4.1). Falls das Verfahren nicht terminiert, entspricht die Ausgabe von Algorithmus 4.1 schließlich dem frühesten W-zulässigen Schedule $ES(W, \alpha, t_\alpha)$.

Im Folgenden wird eine nichtleere Menge $I := \{s, s+1, \ldots, e\} \subset \mathbb{Z}$ genau dann als Startzeitintervall von W_i bezeichnet, wenn die Bedingungen $I \subseteq W_i$ und $s-1, e+1 \notin W_i$ erfüllt sind. Weiterhin wird jedem Zeitpunkt $t \in I$ durch die Funktion $s_i(t)$ der Startzeitpunkt s und durch die Funktion $e_i(t)$ der Endzeitpunkt e des Startzeitintervalls I zugeordnet. Im Fall, dass die Bedingungen $I \subseteq \mathcal{H} \setminus W_i$ und $s-1, e+1 \in W_i$ erfüllt sind, wird I Startzeitunterbrechung von W_i genannt, wobei $\mathcal{B}_i$ und $\mathcal{B}$ jeweils die Anzahl der Startzeitunterbrechungen in W_i und W angeben. Seien I_1 und I_2 unterschiedliche Startzeitintervalle von W_i und gelte $\nu_i \in I_1$ für die Markierung von Vorgang $i \in V$. Dann wird davon gesprochen, dass Vorgang i oder die Markierung ν_i einem anderen Startzeitintervall in W_i zugeordnet wird, falls $\nu_i \in I_2$ nach der Aktualisierung der Markierung ν_i gilt.

Lemma 4.1. *Sei $\nu^\lambda = (\nu_i^\lambda)_{i \in V}$ die Markierung der Vorgänge nach Iteration $\lambda \in \mathbb{Z}_{\geq 0}$ des Algorithmus 4.1. Dann existiert kein zeitzulässiger Schedule, falls nach weiteren $|V|$*

Algorithmus 4.1: Bestimmung des Minimalpunkts von $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha)$

Input: Vorgang $\alpha \in V$ und Zeitpunkt t_α
Output: $ES(W, \alpha, t_\alpha)$

if $\mathcal{S}_T = \emptyset \vee \exists i \in V : W_i = \emptyset \vee t_\alpha > \max W_\alpha$ **then**
  terminate $(\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha) = \emptyset)$
$t'_\alpha := \min \{\tau \in W_\alpha \mid \tau \geq t_\alpha\}$
forall $i \in V$ **do**
  $\nu_i := \begin{cases} t'_\alpha, & \text{falls } i = \alpha \\ -\infty, & \text{sonst} \end{cases}$
$Q := \{\alpha\}$
while $Q \neq \emptyset$ **do**
  Entferne i vom Kopf der Schlange Q
  forall $j \in Succ(i)$ **do**
    if $\nu_i + \widetilde{\delta}_{ij}(W, \nu_i) > \nu_j$ **then**
      $\nu_j := \nu_i + \widetilde{\delta}_{ij}(W, \nu_i)$
      if $j \notin Q$ **then** Füge j am Ende der Schlange Q ein
      if $\nu_j > \max W_j$ **then**
        terminate $(\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha) = \emptyset)$
return $(\nu_i)_{i \in V}$

Iterationen keine Markierung eines Vorgangs $i \in V$ einem anderen Startzeitintervall in W_i zugeordnet wird und die Bedingung $\nu_j \geq \nu_i + \widetilde{\delta}_{ij}(W, \nu_i)$ nicht für jedes Vorgangspaar $(i, j) \in E$ erfüllt ist.

Beweis. Es sei angenommen, dass innerhalb von $|V|$ weiteren Iterationen von Algorithmus 4.1 keine Markierung eines Vorgangs $i \in V$ einem anderen Startzeitintervall zugeordnet wird und nach Iteration $\lambda + |V|$ die Bedingung $\nu_j \geq \nu_i + \widetilde{\delta}_{ij}(W, \nu_i)$ nicht für jedes Vorgangspaar $(i, j) \in E$ erfüllt ist. Zunächst wird in jeder Iteration $\lambda, \lambda + 1, \ldots, \lambda + |V|$ mindestens ein Vorgang der Schlange Q hinzugefügt, sodass nach $|V|$ Iterationen mindestens eine Pfeilfolge $F = \langle i_\lambda, i_{\lambda+1}, \ldots, i_{\lambda+s} \rangle$ mit $s \geq |V|$ in Netzplan N existiert, wobei die Vorgänge mit $\nu_{i_{u+1}} = \nu_{i_u} + \widetilde{\delta}_{i_u, i_{u+1}}(W, \nu_{i_u})$ für alle $u \in \{\lambda, \lambda + 1, \ldots, \lambda + s - 1\}$ markiert wurden. Durch die Annahme, dass über die Iterationen keine Markierung eines Vorgangs einem anderen Startzeitintervall zugeordnet wird, gelten die Bedingungen $\widetilde{\delta}_{i_u, i_{u+1}}(W, \nu_{i_u}) = \delta_{i_u, i_{u+1}}$ für alle $u \in \{\lambda, \lambda + 1, \ldots, \lambda + s - 1\}$. Aus der Länge $s \geq |V|$ der Pfeilfolge F kann schließlich abgeleitet werden, dass Netzplan N mindestens einen Zyklus positiver Länge enthält, sodass $\mathcal{S}_T = \emptyset$ nach Proposition 2.1 in Bartusch et al. (1988) folgt. □

Satz 4.1. *Algorithmus 4.1 bestimmt den eindeutigen Minimalpunkt von* $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha)$ *oder zeigt, dass kein Schedule* $S \in \widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha)$ *existiert, mit einer Zeitkomplexität von* $\mathcal{O}(|V||E|(\mathcal{B}+1))$.

Beweis. Zunächst wird gezeigt, dass die Markierungen der Vorgänge über den Verfahrensverlauf von Algorithmus 4.1 gegen den eindeutigen Minimalpunkt der Menge $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha) \neq \emptyset$ konvergieren. Hierzu sei ein Schedule $S \in \widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha)$ betrachtet. Aus $t'_\alpha := \min\{\tau \in W_\alpha \mid \tau \geq t_\alpha\} \leq S_\alpha$ und $\nu'_i + \widetilde{\delta}_{ij}(W, \nu'_i) \leq \nu''_i + \widetilde{\delta}_{ij}(W, \nu''_i)$ für beliebige Markierungen mit $\nu'_i \leq \nu''_i$ ergibt sich zunächst $\nu_i \leq S_i$ für alle $i \in V$ nach jeder Iteration des Algorithmus 4.1. Da jede weitere Iteration die Erhöhung mindestens einer Markierung ν_i eines Vorgangs $i \in V$ voraussetzt, folgt aus der Annahme $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha) \neq \emptyset$, dass Algorithmus 4.1 mit $Q = \emptyset$ und $\nu := (\nu_i)_{i \in V} \leq S$ nach einer endlichen Anzahl an Iterationen beendet wird. Aus $Q = \emptyset$ am Ende einer Iteration kann schließlich abgeleitet werden, dass die Zeitabstände $\nu_j \geq \nu_i + \widetilde{\delta}_{ij}(W, \nu_i)$ für alle Vorgangspaare $(i, j) \in E$ erfüllt sind, sodass ν nach der letzten Iteration W-zulässig ist (Netzplan N ist stark zusammenhängend). Da $\nu \leq S'$ für jeden Schedule $S' \in \widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha)$ nach der letzten Iteration gilt, stellt ν den eindeutigen Minimalpunkt von $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha)$ dar. Existiert dagegen kein Minimalpunkt bzw. gilt $\widetilde{\mathcal{S}}_T(W, \alpha, t_\alpha) = \emptyset$, terminiert der Algorithmus entweder durch die Bedingungen in Zeile 1 oder nach einer endlichen Anzahl an Iterationen mit $Q \neq \emptyset$.

Die Zeitkomplexität von Algorithmus 4.1 kann wie folgt abgeleitet werden. Zunächst können die Operationen in den Zeilen 1 bis 3 mit einer Zeitkomplexität von $\mathcal{O}(|V||E| + \mathcal{B}_\alpha)$ ausgeführt werden. Dabei wird durch ein Label-Correcting-Verfahren mit einer Zeitkomplexität von $\mathcal{O}(|V||E|)$ überprüft, ob ein Zyklus positiver Länge in Projektnetzplan N vorliegt, woraus genau dann $\mathcal{S}_T = \emptyset$ nach Proposition 2.1 in Bartusch et al. (1988) folgt. Aus Lemma 4.1 ergibt sich eine maximale Anzahl an $\mathcal{O}(|V|(\mathcal{B}+1))$ Iterationen, da nach höchstens $|V|$ Iterationen mindestens ein Vorgang einem anderen Startzeitintervall zugeordnet wird, falls mindestens ein zeitzulässiger Schedule existiert ($\mathcal{S}_T \neq \emptyset$). In jeder Iteration werden höchstens $|E|$ Überprüfungen und mögliche Aktualisierungen der Markierungen durchgeführt. Die Überprüfung der Bedingung $\nu_i + \widetilde{\delta}_{ij}(W, \nu_i) > \nu_j$ kann dabei zunächst durch $\nu_i + \delta_{ij} > \nu_j$ in konstanter Zeit erfolgen. Für den Fall, dass $\nu_i + \delta_{ij}$ im gleichen Startzeitintervall wie ν_j liegt ($\nu_i + \delta_{ij} \leq e_j(\nu_j)$), kann ebenso die Aktualisierung mit einer konstanten Zeitkomplexität durch $\nu_j := \nu_i + \delta_{ij}$ durchgeführt werden. Gilt dagegen $\nu_i + \delta_{ij} > e_j(\nu_j)$, ist zunächst das Startzeitintervall in W_j zu bestimmen, in dem die aktualisierte Markierung von Vorgang j liegt, wozu lediglich die Start- und Endzeitpunkte aller Startzeitintervalle in W_j durchlaufen werden müssen. Für den Verfahrensablauf wird daher die Startzeitbeschränkung W_i jedes Vorgangs $i \in V$ durch eine Liste bestehend aus den Start- und Endzeitpunkten aller Startzeitintervalle verwaltet,

die nach monoton wachsenden Werten sortiert ist. Indem die Endzeitpunkte $e_i(\nu_i)$ und deren Positionen in den jeweiligen Listen für alle Vorgänge $i \in V$ abgespeichert werden, können schließlich die maximal $\mathcal{O}(|V||E|(\mathcal{B}+1))$ Aktualisierungen über alle Iterationen des Algorithmus mit zusätzlich $\mathcal{O}(|V|+\mathcal{B})$ Operationen ausgeführt werden, da jeder Start- und Endzeitpunkt eines Startzeitintervalls maximal einmal durchlaufen wird. □

Das zweite Zeitplanungsverfahren bestimmt den eindeutigen Maximalpunkt der Menge $\widehat{\mathcal{S}}_T(W,\alpha,t_\alpha) := \{S \in \mathcal{S}_T(W) \mid S_\alpha \leq t_\alpha\}$ oder zeigt, dass kein Schedule $S \in \widehat{\mathcal{S}}_T(W,\alpha,t_\alpha)$ existiert. Entsprechend ermittelt das zweite Verfahren die spätestmöglichen W-zulässigen Startzeitpunkte aller Vorgänge des Projekts unter der Annahme, dass ein Vorgang $\alpha \in V$ nicht später als zum Zeitpunkt t_α startet. Mit $\alpha := 0$ und $t_\alpha := 0$ wird der späteste W-zulässige Schedule $LS(W) := \max \mathcal{S}_T(W)$ bestimmt. Der eindeutige Maximalpunkt der Menge $\widehat{\mathcal{S}}_T(W,\alpha,t_\alpha)$ wird im Folgenden mit $LS(W,\alpha,t_\alpha)$ bezeichnet, sodass $LS_i(W,\alpha,t_\alpha)$ dem spätestmöglichen W-zulässigen Startzeitpunkt von Vorgang $i \in V$ unter der Voraussetzung, dass Vorgang $\alpha \in V$ nicht später als zum Zeitpunkt t_α startet, entspricht. Das zweite Zeitplanungsverfahren wird in Algorithmus 4.2 beschrieben. Basierend auf einer Anfangsmarkierung ν mit $\nu_\alpha := \max\{\tau \in W_\alpha \mid \tau \leq t_\alpha\}$ und $\nu_i := \infty$ für alle $i \in V \setminus \{\alpha\}$ werden die Markierungen der Vorgänge iterativ reduziert, bis entweder ein W-zulässiger Schedule bestimmt oder $\widehat{\mathcal{S}}_T(W,\alpha,t_\alpha) = \emptyset$ gezeigt wurde. Abweichend vom ersten Zeitplanungsverfahren wird im Aktualisierungsschritt die Bedingung $\nu_i - \widehat{\delta}_{hi}(W,\nu_i) < \nu_h$ mit

$$\widehat{\delta}_{hi}(W,\nu_i) := \begin{cases} \nu_i - \max\{\tau \in W_h \mid \tau \leq \nu_i - \delta_{hi}\}, & \text{falls } \{\tau \in W_h \mid \tau \leq \nu_i - \delta_{hi}\} \neq \emptyset \\ \delta_{hi}, & \text{sonst} \end{cases}$$

für einen direkten Vorgänger $h \in \mathit{Pred}(i)$ von Vorgang $i \in V$ in Projektnetzplan N überprüft. Der Zeitpunkt $\nu_i - \widehat{\delta}_{hi}(W,\nu_i)$ entspricht dabei dem spätestmöglichen Startzeitpunkt $\nu_h \in W_h$ von Vorgang h, der den Zeitabstand $\nu_h \leq \nu_i - \delta_{hi}$ einhält (falls $\nu_i - \delta_{hi} \geq \min W_h$).

Satz 4.2. *Algorithmus 4.2 bestimmt den eindeutigen Maximalpunkt von $\widehat{\mathcal{S}}_T(W,\alpha,t_\alpha)$ oder zeigt, dass kein Schedule $S \in \widehat{\mathcal{S}}_T(W,\alpha,t_\alpha)$ existiert, mit einer Zeitkomplexität von $\mathcal{O}(|V||E|(\mathcal{B}+1))$.*

Beweis. Analog zum Beweis von Satz 4.1. □

Es ist zu beachten, dass das Problem (PS(W)) einem Spezialfall des in Franck (1999) beschriebenen Projektdauerminimierungsproblems mit allgemeinen Zeitbeziehungen, Kalendern und unbeschränkt zur Verfügung stehenden Ressourcen entspricht. Die Zeitpla-

Algorithmus 4.2: Bestimmung des Maximalpunkts von $\widehat{\mathcal{S}}_T(W, \alpha, t_\alpha)$

Input: Vorgang $\alpha \in V$ und Zeitpunkt t_α
Output: $LS(W, \alpha, t_\alpha)$

if $\mathcal{S}_T = \emptyset \vee \exists i \in V : W_i = \emptyset \vee t_\alpha < \min W_\alpha$ **then**
  terminate $(\widehat{\mathcal{S}}_T(W, \alpha, t_\alpha) = \emptyset)$
$t'_\alpha := \max\{\tau \in W_\alpha \mid \tau \leq t_\alpha\}$
forall $i \in V$ **do**
  $\nu_i := \begin{cases} t'_\alpha, & \text{falls } i = \alpha \\ \infty, & \text{sonst} \end{cases}$
$Q := \{\alpha\}$
while $Q \neq \emptyset$ **do**
  Entferne i vom Kopf der Schlange Q
  forall $h \in Pred(i)$ **do**
    if $\nu_i - \widehat{\delta}_{hi}(W, \nu_i) < \nu_h$ **then**
      $\nu_h := \nu_i - \widehat{\delta}_{hi}(W, \nu_i)$
      if $h \notin Q$ **then** Füge h am Ende der Schlange Q ein
      if $\nu_h < \min W_h$ **then**
        terminate $(\widehat{\mathcal{S}}_T(W, \alpha, t_\alpha) = \emptyset)$
return $(\nu_i)_{i \in V}$

nungsverfahren, die in Franck (1999, Abschnitt 3.3.3) und Franck et al. (2001a) beschrieben werden, können daher ebenfalls zur Bestimmung von $ES(W, \alpha, t_\alpha)$ und $LS(W, \alpha, t_\alpha)$ verwendet werden. Beide Verfahren weisen dabei eine Zeitkomplexität von $\mathcal{O}(|V||E|(\mathcal{B}+1))$ in Bezug auf die Notation der vorliegenden Arbeit auf.

4.1.2 Mindest- und Höchstabstände

Die folgenden Zeitplanungsverfahren bestimmen alle W-zulässigen Startzeitpunkte der Vorgänge eines Projekts sowie die indirekten zeitlichen Mindest- und Höchstabstände zwischen den Startzeitpunkten der Vorgänge unter Berücksichtigung einer vorgegebenen Startzeitbeschränkung W. Der indirekte zeitliche Mindestabstand (Höchstabstand) $\widetilde{d}_{ij}(W, t)$ $(\widehat{d}_{ij}(W, t))$ eines Vorgangspaars $(i, j) \in V \times V$ entspricht dabei dem Zeitabstand zwischen Zeitpunkt t und dem frühestmöglichen (spätestmöglichen) W-zulässigen Startzeitpunkt von Vorgang j, unter der Voraussetzung, dass Vorgang i nicht früher (später) als zum Zeitpunkt t startet. Entsprechend ist der indirekte zeitliche Mindestabstand (Höchstabstand) durch $\widetilde{d}_{ij}(W, t) := ES_j(W, i, t) - t$ $(\widehat{d}_{ij}(W, t) := LS_j(W, i, t) - t)$ definiert, sodass jeder W-zulässige Startzeitpunkt eines Vorgangs j im planungsabhängigen Startzeitfenster $[t + \widetilde{d}_{ij}(W, t), t + \widehat{d}_{ij}(W, t)]$ liegt, falls Vorgang i genau zum Zeitpunkt t

startet. Im Folgenden werden zwei Zeitplanungsverfahren vorgestellt, die für jedes Vorgangspaar $(i,j) \in V \times V$, $i \neq j$ und jeden Zeitpunkt t einer Teilmenge aller Zeitpunkte des Planungshorizonts die indirekten zeitlichen Mindest- und Höchstabstände $\widetilde{d}_{ij}(W,t)$ und $\widehat{d}_{ij}(W,t)$ bestimmen, mit denen die indirekten zeitlichen Mindest- und Höchstabstände für alle W-zulässigen Startzeitpunkte von Vorgang i berechnet werden können.

Algorithmus 4.3: Bestimmung der indirekten zeitlichen Mindestabstände

Input: Startzeitbeschränkung W
Output: Mindestabstandsmatrix $\widetilde{D}(W)$

if $\mathcal{S}_T = \emptyset \vee \exists i \in V : W_i = \emptyset$ **then**
  terminate $(\mathcal{S}_T(W) = \emptyset)$
forall $i \in V$ **do**
  $t := \min W_i$
  while $\widetilde{\mathcal{S}}_T(W,i,t) \neq \emptyset$ **do**
    $ES' := ES(W,i,t)$
    if $ES'_i > t$ **then**
      $W_i := W_i \setminus [t, ES'_i - 1]$
      $t := ES'_i$
    forall $j \in V \setminus \{i\}$ **do**
      $d_{ijt} := ES'_j - ES'_i$
    $t' := \min(\min_{j \in V \setminus \{i\}}(e_j(ES'_j) - d_{ij}), -d_{i0})$
    if $t' > t$ **then**
      forall $j \in V \setminus \{i\}$ **do**
        if $ES'_j - d_{ij} < t'$ **then**
          if $t < ES'_j - d_{ij}$ **then**
            $d_{ij,ES'_j - d_{ij}} := d_{ij}$
          $d_{ijt'} := d_{ij}$
        else
          $d_{ijt'} := ES'_j - t'$
    $t := t' + 1$
  $W_i := W_i \setminus [t, \infty)$
return $\widetilde{D}(W) = ([d_{ijt}])_{i,j \in V}$

Das Verfahren zur Bestimmung aller W-zulässigen Startzeitpunkte sowie zur Berechnung der indirekten zeitlichen Mindestabstände $\widetilde{d}_{ij}(W,t)$ für alle Vorgangspaare $(i,j) \in V \times V$, $i \neq j$ und jeden Zeitpunkt t einer Teilmenge aller Zeitpunkte des Planungshorizonts wird in Algorithmus 4.3 beschrieben. Die indirekten zeitlichen Mindestabstände $\widetilde{d}_{ij}(W,t)$ werden in einer nach nichtfallenden Werten von t sortierten Liste $[\widetilde{d}_{ij}(W,t)]$ abgespeichert und in der Mindestabstandsmatrix $\widetilde{D}(W) := ([\widetilde{d}_{ij}(W,t)])_{i,j \in V}$ zusammengefasst. Algorithmus 4.3 basiert auf einer schrittweisen Erhöhung der Zeitpunkte innerhalb einer

Startzeitbeschränkung W_i, beginnend mit dem kleinsten Zeitpunkt $t = \min W_i$. Solange für einen betrachteten Vorgang $i \in V$ die Bedingung $\widetilde{S}_T(W, i, t) \neq \emptyset$ erfüllt ist, wird der eindeutige Minimalpunkt $ES' := ES(W, i, t)$ bestimmt. Falls $ES'_i > t$ gilt, folgt aus Satz 4.1, dass alle Zeitpunkte $\tau \in \{t, t+1, \ldots, ES'_i - 1\}$ nicht W-zulässig sind, sodass sie in Zeile 8 aus W_i entfernt werden. Nachdem die Variable t auf ES'_i gesetzt wurde, werden die indirekten zeitlichen Mindestabstände für Zeitpunkt t und jeden Vorgang $j \in V \setminus \{i\}$ durch die Variable $d_{ijt} := ES'_j - ES'_i$ abgespeichert. Im Anschluss daran wird ein Zeitpunkt t' berechnet, bis zu dem der Zeitpunkt t erhöht werden kann, sodass alle Startzeitpunkte $\tau \in [t, t'] \cap W_i$ nach Satz 4.3 W-zulässig sind.

Satz 4.3. *Gegeben seien* $ES' = \min \widetilde{\mathcal{S}}_T(W, i, t)$ *und* $ES'' = \min \widetilde{\mathcal{S}}_T(W, i, \tau)$. *Dann gilt für jeden Zeitpunkt* $\tau \in W_i$ *mit* $ES'_i \leq \tau \leq \min(\min_{j \in V \setminus \{i\}}(e_j(ES'_j) - d_{ij}), -d_{i0})$ *die Bedingung* $ES''_h = \max(ES'_h, \tau + d_{ih})$ *für jeden Vorgang* $h \in V$.

Beweis. Gegeben sei ein Schedule $S = (S_h)_{h \in V}$ mit $S_h = \max(ES'_h, \tau + d_{ih})$. Zunächst kann aus $\widetilde{\mathcal{S}}_T(W, i, t) \supseteq \widetilde{\mathcal{S}}_T(W, i, \tau)$ abgeleitet werden, dass $ES'_h \leq ES''_h$ für jeden Vorgang $h \in V$ gilt. Da weiterhin d_{ih} einer unteren Schranke für den Zeitabstand $ES''_h - \tau$ entspricht, gilt $(\tau + d_{ih})_{h \in V} \leq ES''$ und somit $S \leq ES''$. Durch $S_i = \max(ES'_i, \tau + d_{ii}) = \tau$ ($d_{ii} = 0$) reicht es schließlich aus, die W-Zulässigkeit von S zu zeigen, um $S = ES''$ zu beweisen.

Zunächst wird die Zeitzulässigkeit von S gezeigt. Da $S_0 = 0$ aus den Bedingungen $ES'_0 = 0$ und $\tau \leq -d_{i0}$ ($\tau + d_{i0} \leq 0$) folgt, muss mindestens ein Vorgangspaar $(u, v) \in V \times V$ mit $S_v < S_u + d_{uv}$ existieren, falls S nicht zeitzulässig ist. Durch die Zeitzulässigkeit von ES' ist der indirekte Zeitabstand $ES'_v \geq ES'_u + d_{uv}$ zwischen dem Vorgangspaar erfüllt, woraus sich $S_u = \tau + d_{iu} > ES'_u$ ableiten lässt. Es ergibt sich ein Widerspruch zur Annahme $S_v = \max(ES'_v, \tau + d_{iv})$ durch $S_v < \tau + d_{iu} + d_{uv} \leq \tau + d_{iv}$, woraus die Zeitzulässigkeit von S folgt. Da $\tau \in W_i$ gilt und $\tau \leq \min_{h \in V \setminus \{i\}}(e_h(ES'_h) - d_{ih})$ die Bedingung $S_h \in W_h$ für jeden Vorgang $h \in V \setminus \{i\}$ sicherstellt, ist S schließlich W-zulässig. □

Für den Fall $t' > t$ werden im nächsten Schritt des Verfahrens die Zeitpunkte und die zugehörigen zeitlichen Mindestabstände ermittelt, die in den Listen $[\widetilde{d}_{ij}(W, t)]$ abgespeichert werden. Zur Bestimmung der entsprechenden Zeitpunkte kann aus Satz 4.3 zunächst abgeleitet werden, dass die Bedingungen $\widetilde{d}_{ij}(W, \tau') - (\tau'' - \tau') = \widetilde{d}_{ij}(W, \tau'')$ für alle Zeitpunkte $\tau', \tau'' \in [t, t'] \cap W_i$ mit $\tau' < \tau''$ und $\tau'' + d_{ij} \leq ES'_j$ gelten. Weiterhin ergibt sich aus Satz 4.3, dass alle anderen Zeitpunkte, für die $\tau' + d_{ij} \geq ES'_j$ gilt, die Bedingungen $\widetilde{d}_{ij}(W, \tau') = \widetilde{d}_{ij}(W, \tau'') = d_{ij}$ erfüllen. Schließlich folgt daraus, dass es ausreichend ist, die indirekten zeitlichen Mindestabstände $\widetilde{d}_{ij}(W, ES'_j - d_{ij})$ für den Fall $ES'_j - d_{ij} < t'$ und $\widetilde{d}_{ij}(W, t')$ abzuspeichern, um $\widetilde{d}_{ij}(W, \tau)$ für jeden Zeitpunkt $\tau \in [t, t'] \cap W_i$ berechnen

zu können. Die Operationen zum Abspeichern der jeweiligen Werte in $[\widetilde{d}_{ij}(W,t)]$ werden in den Zeilen 13 bis 20 von Algorithmus 4.3 beschrieben. Der indirekte zeitliche Mindestabstand $\widetilde{d}_{ij}(W,\tau)$ für jeden W-zulässigen Startzeitpunkt τ im Intervall $[t,t']$ kann schließlich basierend auf Satz 4.3 durch

$$\widetilde{d}_{ij}(W,\tau) := \widetilde{d}_{ij}(W,\tau') + \operatorname{sgn}\left(\widetilde{d}_{ij}(W,\tau'') - \widetilde{d}_{ij}(W,\tau')\right) \cdot (\tau - \tau') \tag{4.1}$$

mit $\tau' := \max\{\sigma \in \Psi \mid \sigma \leq \tau\}$, $\tau'' := \min\{\sigma \in \Psi \mid \sigma \geq \tau\}$ und Ψ als Menge aller abgespeicherten Zeitpunkte in $[\widetilde{d}_{ij}(W,t)]$ bestimmt werden. Für den Fall, dass nach einer Iteration für Vorgang i weiterhin ein W-zulässiger Startzeitpunkt $\tau \geq t'+1$ in W_i existiert, wird eine weitere Iteration für Vorgang i ausgeführt. Andernfalls werden alle verbleibenden Zeitpunkte $\tau \geq t'+1$ aus W_i entfernt und der nächste Vorgang betrachtet. Nach Ablauf des Algorithmus 4.3 werden schließlich die Mindestabstandsmatrix $\widetilde{D}(W)$ und die Startzeitbeschränkung W, die ausschließlich alle W-zulässigen Startzeitpunkte der anfänglichen Startzeitbeschränkung enthält, ausgegeben.

Lemma 4.2. *Gegeben seien ein Schedule $ES' = \min \widetilde{\mathcal{S}}_T(W,\alpha,t'_\alpha)$ und eine Markierung $\nu = (\nu_i)_{i\in V}$ mit $\nu_i = ES'_i$ für alle $i \in V \setminus \{\alpha\}$ und $\nu_\alpha = \min\{\tau \in W_\alpha \mid \tau \geq t''_\alpha\}$ mit $ES'_\alpha < t''_\alpha \leq \max W_\alpha$. Dann bestimmt Algorithmus 4.1 den eindeutigen Minimalpunkt $ES'' = \min \widetilde{\mathcal{S}}_T(W,\alpha,t''_\alpha)$ oder zeigt, dass ES'' nicht existiert, wenn das Verfahren mit $Q = \{\alpha\}$ und Markierung ν initialisiert wird.*

Beweis. Zunächst kann $\nu \leq ES''$ aus $\widetilde{\mathcal{S}}_T(W,\alpha,t''_\alpha) \subset \widetilde{\mathcal{S}}_T(W,\alpha,t'_\alpha)$ und $\nu_\alpha = \min\{\tau \in W_\alpha | \tau \geq t''_\alpha\}$ abgeleitet werden. Weiterhin kann festgestellt werden, dass alle Bedingungen $\nu_j \geq \nu_i + \widetilde{\delta}_{ij}(W,\nu_i)$ für alle $(i,j) \in E$ mit $i \neq \alpha$ erfüllt sind, und dass die Initialisierung $Q = \{\alpha\}$ die Überprüfung der Bedingungen $\nu_j \geq \nu_\alpha + \widetilde{\delta}_{\alpha j}(W,\nu_\alpha)$ für alle $j \in Succ(\alpha)$ in der ersten Iteration sicherstellt. Da $\nu_\alpha \geq t''_\alpha$ mit $\nu_\alpha \in W_\alpha$ gilt, folgt analog zum Beweis von Satz 4.1, dass Algorithmus 4.1 entweder den Minimalpunkt ES'' bestimmt oder zeigt, dass kein Schedule $S \in \widetilde{S}_T(W,\alpha,t''_\alpha)$ existiert. □

Lemma 4.3. *Sei $\nu^\lambda = (\nu_i^\lambda)_{i\in V}$ eine Markierung und Q^λ die zugehörige Vorgangsmenge nach λ Iterationen von Algorithmus 4.1. Dann gilt für die Markierung $\nu_i^{\lambda+u}$ eines Vorgangs $i \in V$ nach u weiteren Iterationen $\nu_i^{\lambda+u} \geq \max_{q\in Q^\lambda}(\nu_q^\lambda + d_{qi}^u)$ mit d_{qi}^u als Länge einer längsten Pfeilfolge von q nach i in Netzplan N mit maximal u Pfeilen.*

Beweis. Der nachfolgende Beweis erfolgt mit Hilfe vollständiger Induktion. Zunächst gilt der Induktionsanfang für $u = 1$, da die Bedingungen $\nu_i^{\lambda+1} \geq \max(\nu_i^\lambda, \max_{q\in Q^\lambda \cap Pred(i)}(\nu_q^\lambda + d_{qi}^1))$ mit $d_{qi}^1 = \delta_{qi}$ erfüllt sind. Weiterhin ist die Induktionsannahme nach $u+1$ $(u > 1)$ weiteren Iterationen für jeden Vorgang $i \in V$ erfüllt, wenn alle längsten Pfeilfolgen von

jedem Vorgang $q \in Q^\lambda$ zu Vorgang i aus weniger als $u+1$ Pfeilen bestehen oder $\nu_i^{\lambda+u}$ bereits größer als $\nu_q^\lambda + d_{qi}^{u+1}$ für alle $q \in Q^\lambda$ ist. Andernfalls, wenn also eine längste Pfeilfolge in Netzplan N von mindestens einem Vorgang $q \in Q^\lambda$ zu Vorgang i mit $u+1$ und nicht weniger Pfeilen existiert und $\nu_i^{\lambda+u} < \nu_q^\lambda + d_{qi}^{u+1}$ gilt, folgt aus dem Aktualisierungsschritt in Algorithmus 4.1 und der Induktionsannahme

$$\nu_i^{\lambda+u+1} \geq \nu_h^{\lambda+u} + \delta_{hi} \geq \nu_q^\lambda + d_{qh}^u + \delta_{hi} = \nu_q^\lambda + d_{qi}^{u+1}$$

mit h als dem direkten Vorgänger von i auf einer längsten Pfeilfolge von q nach i in Netzplan N mit $u+1$ Pfeilen. □

Lemma 4.4. *Gegeben seien zwei Startzeitbeschränkungen W' und W'' mit $W_i' \supseteq W_i''$ für alle $i \in V$, wobei W_i'' alle W'-zulässigen Startzeitpunkte enthält. Weiterhin sei angenommen, dass Algorithmus 4.1 bereits für λ Iterationen auf Startzeitbeschränkung W'' ausgeführt wurde. Dann existiert kein zeitzulässiger Schedule, falls Algorithmus 4.1 nicht innerhalb von $|V|$ weiteren Iterationen endet und keine Markierung ν_i eines Vorgangs $i \in V$ einem anderen Startzeitintervall von W_i' zugeordnet wird.*

Beweis. Für den Beweis sei angenommen, dass mindestens ein zeitzulässiger Schedule existiert, wodurch es ausreicht zu zeigen, dass Algorithmus 4.1 unter den genannten Bedingungen nach höchstens $|V|$ weiteren Iterationen endet. Die Markierung und die Vorgangsmenge nach Iteration λ seien durch $\nu^\lambda = (\nu_i^\lambda)_{i \in V}$ und Q^λ gegeben, wobei $\alpha \in V$ und t_α dem Input von Algorithmus 4.1 entspricht.

Gegeben sei die Markierung $\nu' = (\nu_i')_{i \in V}$ mit $\nu_i' = \max(\nu_i^\lambda, \max_{q \in Q^\lambda}(\nu_q^\lambda + d_{qi}))$. Aus Lemma 4.3 folgt, dass die Markierung von Vorgang $i \in V$ nach $|V|-1$ Iterationen mindestens so groß wie ν_i' ist. Falls mindestens ein Vorgang $i \in V$ mit $\nu_i' > \max W_i' \geq \max W_i''$ existiert, wird der Algorithmus dementsprechend nach höchstens $|V|$ weiteren Iterationen beendet, sodass im Folgenden nur noch der Fall $\nu_i' \leq \max W_i'$ für alle Vorgänge $i \in V$ betrachtet wird. Da keine Markierung eines Vorgangs i einem anderen Startzeitintervall von W_i' zugeordnet wird und $\nu_i^\lambda \in W_i'$ für alle Vorgänge $i \in V$ durch $W_i' \supseteq W_i''$ gilt, folgt $\nu_i' \in W_i'$ für alle Vorgänge $i \in V$. Im Folgenden wird die Zeitzulässigkeit von ν' bewiesen, woraus sich die W-Zulässigkeit für die Startzeitbeschränkung W' und somit gemäß der Annahmen des Lemmas auch für W'' ergibt. Es wird angenommen, dass ν' (mit $\nu_0' = 0$) nicht zeitzulässig ist. In diesem Fall existiert mindestens ein Vorgangspaar $(i,j) \in E$ mit $\nu_j' < \nu_i' + \delta_{ij}$. Zunächst wird der Fall $\nu_i' > \nu_i^\lambda$ betrachtet, sodass $\nu_i' = \nu_q^\lambda + d_{qi}$ für mindestens einen Vorgang $q \in Q^\lambda$ gilt. Da aus $\nu_j' < \nu_q^\lambda + d_{qi} + \delta_{ij} \leq \nu_q^\lambda + d_{qj}$ jedoch ein Widerspruch zur Berechnungsvorschrift von ν' folgt, sind alle Zeitbeziehungen, für die $\nu_i' > \nu_i^\lambda$ gilt, erfüllt. Schließlich wird der Fall $\nu_i' = \nu_i^\lambda$ betrachtet. Falls $i \in Q^\lambda$ gilt, dann

ist $\nu'_j \geq \nu^\lambda_i + d_{ij} \geq \nu'_i + \delta_{ij}$ durch die Berechnungsvorschrift von ν' sichergestellt. Andernfalls ergibt sich aus $i \notin Q^\lambda$ die Bedingung $\nu^\lambda_j \geq \nu^\lambda_i + \delta_{ij}$, woraus $\nu'_j \geq \nu'_i + \delta_{ij}$ und somit die Zeitzulässigkeit von ν' folgt. Da $\nu' \in \widetilde{\mathcal{S}}_T(W'', \alpha, t_\alpha)$ gilt, stellt ν'_i eine obere Schranke für jede Markierung ν^u_i eines Vorgangs i nach einer beliebigen Iteration u des Algorithmus 4.1 dar (s. Beweis zu Satz 4.1). Daraus folgt schließlich, dass der Algorithmus 4.1 nach höchstens $|V|$ weiteren Iterationen endet, da nach Lemma 4.3 die Markierung eines Vorgangs $i \in V$ nach $|V| - 1$ weiteren Iterationen mindestens so groß wie ν'_i ist. □

Satz 4.4. *Algorithmus 4.3 bestimmt für jeden Vorgang $i \in V$ alle W-zulässigen Startzeitpunkte und den indirekten zeitlichen Mindestabstand $\widetilde{d}_{ij}(W, t)$ zu jedem Vorgang $j \in V \setminus \{i\}$ für eine Teilmenge aller Zeitpunkte $t \in \mathcal{H}$, durch die der zeitliche Mindestabstand $\widetilde{d}_{ij}(W, \tau)$ für jeden W-zulässigen Startzeitpunkt $\tau \in W_i$ bestimmt werden kann, mit einer Zeitkomplexität von $\mathcal{O}(|V|^2|E|(\mathcal{B}+1))$.*

Beweis. Die Korrektheit des Algorithmus 4.3 folgt aus den Beschreibungen des Verfahrens und den Sätzen 4.1 und 4.3, sodass im Folgenden nur noch die Zeitkomplexität betrachtet wird. Die Terminierungsbedingungen in Zeile 1 werden zunächst mit einer Zeitkomplexität von $\mathcal{O}(|V||E|)$ überprüft. Weiterhin kann gezeigt werden, dass für jeden Vorgang $i \in V$ maximal $\mathcal{O}(\mathcal{B}+1)$ Iterationen (Schleifendurchläufe beginnend in Zeile 5) ausgeführt werden. Dafür wird im Folgenden die Startzeitbeschränkung vor der ersten Iteration für einen Vorgang $i \in V$, d. h., wenn alle W-unzulässigen Startzeitpunkte für eine Teilmenge der Vorgänge $j \in V' \subseteq V \setminus \{i\}$ in vorangegangenen Iterationen bereits eliminiert wurden, mit W^i bezeichnet und die Anzahl der Startzeitunterbrechungen durch $\mathcal{B}^i$ angegeben. W und $\mathcal{B}$ geben jeweils die Startzeitbeschränkung und die zugehörige Anzahl der Startzeitunterbrechungen zu Beginn von Algorithmus 4.3 an. Weiterhin bezeichnet ES'^+ den frühesten W-zulässigen Schedule $ES(W^i, i, t'+1)$ am Ende einer betrachteten Iteration. Zunächst lässt sich aus $t'+1 \notin W^i_i$ am Ende einer Iteration (Zeile 21) aus $W^i_i = W_i$ ableiten, dass entweder $\widetilde{\mathcal{S}}_T(W^i, i, t'+1) = \emptyset$ gilt oder ES'^+_i einer anderen Startzeitkomponente in W_i als ES'_i zugeordnet ist. Weiterhin gilt für $t'+1 \in W^i_i$ mit $\widetilde{\mathcal{S}}_T(W^i, i, t'+1) \neq \emptyset$ am Ende einer Iteration, dass der Zeitpunkt ES'^+_j für mindestens einen Vorgang $j \in V \setminus \{i\}$ einer anderen Startzeitkomponente in W_j als ES'_j zugeordnet ist. Diese Aussage kann aus Satz 4.3 abgeleitet werden, da aus der Annahme, dass der Zeitpunkt ES'^+_j für keinen Vorgang $j \in V$ einer anderen Startzeitkomponente in W_j als ES'_j zugeordnet ist, $ES(W, i, t'+1) = (\max(ES'_j, t'+1+d_{ij}))_{j \in V} = ES'^+$ folgt, woraus sich entweder ein Widerspruch zur Berechnungsvorschrift für t' in Zeile 12 oder zur Korrektheit des Algorithmus ergibt. Entsprechend werden maximal $\mathcal{O}(\mathcal{B}+1)$ Iterationen für jeden Vorgang $i \in V$ ausgeführt. In jeder Iteration für einen Vorgang $i \in V$ können, abgesehen von den Zeilen 5, 6 und 8, alle Operationen mit einer Zeitkomplexität von $\mathcal{O}(|V|)$

durchgeführt werden, wobei $e_j(ES'_j)$ für jeden Vorgang $j \in V \setminus \{i\}$ in Algorithmus 4.1 abgespeichert und zur Berechnung von t' in Zeile 12 verwendet wird. Die Operationen für die Entfernung nicht W-zulässiger Startzeitpunkte aus W_i in den Zeilen 8 und 22 können über alle Iterationen mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{B}_i)$ durchgeführt werden ($W_i^i = W_i$). Abschließend wird die Zeitkomplexität zur Bestimmung von ES' über alle Iterationen für einen Vorgang $i \in V$ betrachtet. Aus Lemma 4.2 folgt zunächst, dass in jeder Iteration von Algorithmus 4.3 der Algorithmus 4.1 mit den Ergebnissen einer vorherigen Iteration initialisiert werden kann, sodass die Markierungen der Vorgänge sowie die zugehörigen Startzeitintervalle abgespeichert und in einer nächsten Iteration wieder verwendet werden können. Da in jeder Iteration für einen Vorgang $i \in V$ in Algorithmus 4.3 für $\widetilde{\mathcal{S}}_T(W^i, i, t'+1) \neq \emptyset$ der Zeitpunkt $ES_j'^+$ mindestens eines Vorgangs $j \in V$ einem anderen Startzeitintervall in W_j als ES'_j zugeordnet ist (s. o.), kann zunächst analog zum Beweis von Satz 4.1 eine maximale Anzahl an $\mathcal{O}(|V|(\mathcal{B}^i+1))$ Iterationen für Algorithmus 4.1 über alle Iterationen für Vorgang $i \in V$ in Algorithmus 4.3 abgeleitet werden. Schließlich folgt aus Lemma 4.4 eine maximale Anzahl an $\mathcal{O}(|V|(\mathcal{B}+1))$ Iterationen für Algorithmus 4.1 über alle Iterationen für Vorgang $i \in V$ in Algorithmus 4.3, sodass höchstens $\mathcal{O}(|V||E|(\mathcal{B}+1))$ Überprüfungen und Aktualisierungen der Vorgangsmarkierungen ausgeführt werden. Unter Berücksichtigung, dass eine Startzeitbeschränkung W^i maximal $|V|\mathcal{B}$ Startzeitunterbrechungen aufweist (s. Zeile 8), folgt analog zum Beweis von Satz 4.1 eine Zeitkomplexität von $\mathcal{O}(|V|\mathcal{B}+|V|)$ für die Aktualisierung der Markierungen über alle Iterationen von Algorithmus 4.1 für Vorgang $i \in V$ in Algorithmus 4.3. Zusammenfassend ergibt sich eine Zeitkomplexität von $\mathcal{O}(|V|^2|E|(\mathcal{B}+1))$ für Algorithmus 4.3. □

Um die Vorgehensweise von Algorithmus 4.3 zu veranschaulichen, zeigt Abbildung 4.2 den Ausschnitt eines Projektnetzplans N mit drei Vorgängen. Zur Vereinfachung werden alle weiteren Vorgänge des Projekts im Folgenden vernachlässigt. Die Startzeitbeschränkungen der Vorgänge, die in der Abbildung durch schraffierte Flächen für die Vorgänge 2 und 3 angedeutet werden, sind durch $W_1 = \{0, 1, \ldots, 9\}$, $W_2 = \{3, 4, 5, 6, 7, 10, 11\}$ und $W_3 = \{4, 5, 6, 7, 11, 12\}$ gegeben. Abbildung 4.2 skizziert den Verlauf der ersten Iteration von Algorithmus 4.3 für Vorgang $i = 1$, wobei nachfolgend nur die zeitlichen Mindestabstände zu Vorgang $j = 3$ betrachtet werden. Zunächst werden für den Zeitpunkt $t := \min W_1 = 0$ die frühesten W-zulässigen Startzeitpunkte der Vorgänge mit $ES'_1 = ES_1(W, 1, 0) = 0$, $ES'_2 = ES_2(W, 1, 0) = 3$ und $ES'_3 = ES_3(W, 1, 0) = 4$ bestimmt. Anschließend wird der indirekte zeitliche Mindestabstand $d_{1,3,0} := ES'_3 - ES'_1 = 4 - 0 = 4$ gesetzt und der Zeitpunkt $t' = \min(\min(e_2(ES'_2) - d_{12}, e_3(ES'_3) - d_{13}), -d_{10}) = \min(\min(7-3, 7-2), 8) = 4$ unter der Annahme $LS_1 = -d_{10} = 8$ berechnet. Da $ES'_3 - d_{13} = 2 < t' = 4$ gilt, ergibt sich für Vorgang $i = 1$ zunächst eine Erhöhung des Zeitpunkts $t = 0$ um zwei Zeiteinheiten

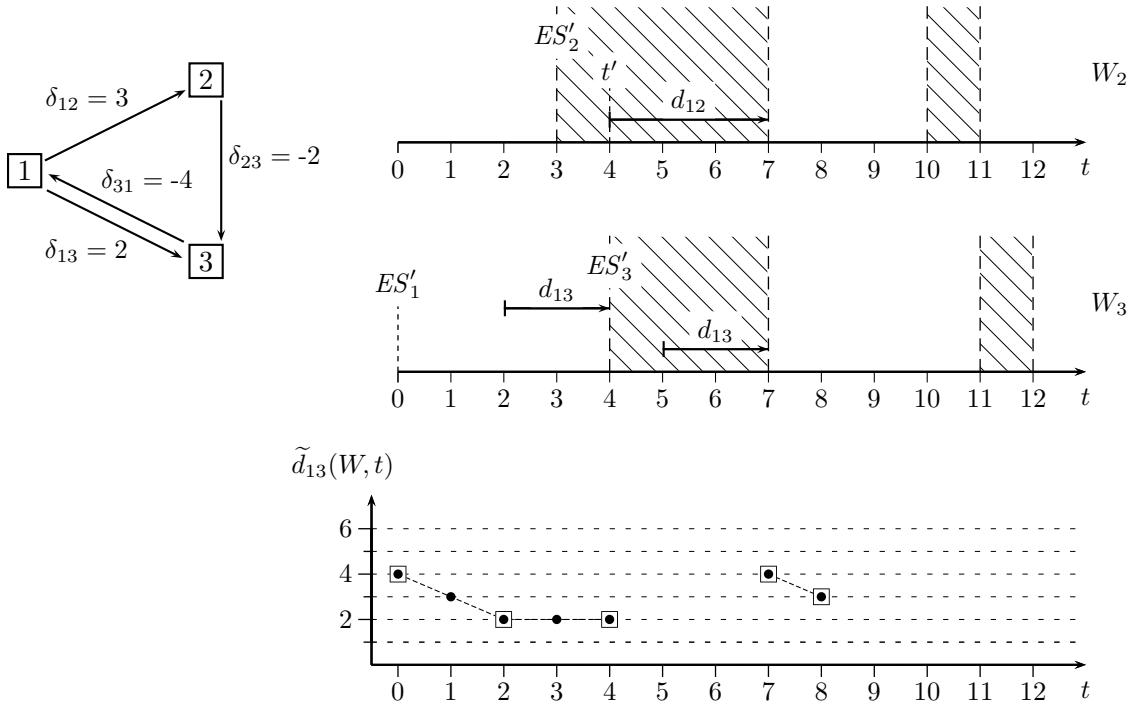

Abbildung 4.2: Indirekte zeitliche Mindestabstände

bis Zeitpunkt $ES'_3 - d_{13} = 2$ erreicht wird, sodass die Zeitabstände $d_{1,3,2} = d_{13} = 2$ und $d_{1,3,4} = d_{13} = 2$ abgespeichert werden. In der nächsten Iteration mit $t := t' + 1 = 5$ werden die frühesten W-zulässigen Startzeitpunkte der Vorgänge mit $ES'_1 = ES_1(W, 1, 5) = 7$, $ES'_2 = ES_2(W, 1, 5) = 10$ und $ES'_3 = ES_3(W, 1, 5) = 11$ bestimmt, sodass die Startzeitbeschränkung von Vorgang $i = 1$ durch $W_1 := W_1 \setminus [5, 6] = \{0, 1, \ldots, 4, 7, 8, 9\}$ aktualisiert und Zeitpunkt $t := ES'_1 = 7$ gesetzt wird. Die Startzeitpunkte $t = 5$ und $t = 6$ sind dementsprechend W-unzulässig. Das untere Diagramm in Abbildung 4.2 zeigt die indirekten zeitlichen Mindestabstände zwischen den Vorgängen $i = 1$ und $j = 3$ für alle W-zulässigen Startzeitpunkte von Vorgang $i = 1$ nach Ablauf von Algorithmus 4.3. Die Zeitabstände, die in der Liste $[\widetilde{d}_{13}(W, t)]$ im Verlauf von Algorithmus 4.3 abgespeichert wurden, sind durch Quadrate im unteren Diagramm der Abbildung gekennzeichnet. Weiterhin sind die Zeitabstände, die innerhalb einer Iteration abgespeichert wurden, durch gestrichelte Linien miteinander verbunden, um die Bestimmung der Zeitabstände für alle W-zulässigen Startzeitpunkte von Vorgang $i = 1$ durch die Berechnungsvorschrift (4.1) zu verdeutlichen.

Das Zeitplanungsverfahren, das in Algorithmus 4.4 beschrieben wird, bestimmt für jedes Vorgangspaar $(i, j) \in V \times V$, $i \neq j$ eine nach nichtfallenden Werten von t sortierte Liste $[\widehat{d}_{ij}(W, t)]$ für eine Teilmenge aller Zeitpunkte $t \in \mathcal{H}$, um den indirekten zeitlichen Höchstabstand $\widehat{d}_{ij}(W, \tau)$ für jeden W-zulässigen Startzeitpunkt $\tau \in W_i$ eines Vorgangs $i \in$

Algorithmus 4.4: Bestimmung der indirekten zeitlichen Höchstabstände

Input: Startzeitbeschränkung W
Output: Höchstabstandsmatrix $\widehat{D}(W)$

if $\mathcal{S}_T = \emptyset \vee \exists i \in V : W_i = \emptyset$ **then**
  terminate ($\mathcal{S}_T(W) = \emptyset$)
forall $i \in V$ **do**
  $t := \max W_i$
  while $\widehat{\mathcal{S}}_T(W, i, t) \neq \emptyset$ **do**
    $LS' := LS(W, i, t)$
    if $LS'_i < t$ **then**
      $W_i := W_i \setminus [LS'_i + 1, t]$
      $t := LS'_i$
    forall $j \in V \setminus \{i\}$ **do**
      $d_{ijt} := LS'_j - LS'_i$
    $t' := \max(\max_{j \in V \setminus \{i\}}(s_j(LS'_j) + d_{ji}), d_{0i})$
    if $t' < t$ **then**
      forall $j \in V \setminus \{i\}$ **do**
        if $LS'_j + d_{ji} > t'$ **then**
          if $t > LS'_j + d_{ji}$ **then**
            $d_{ij,LS'_j+d_{ji}} := -d_{ji}$
          $d_{ijt'} := -d_{ji}$
        else
          $d_{ijt'} := LS'_j - t'$
    $t := t' - 1$
  $W_i := W_i \setminus [0, t]$
return $\widehat{D}(W) = ([d_{ijt}])_{i,j \in V}$

V ermitteln zu können. Zusätzlich bestimmt das Verfahren, wie auch Algorithmus 4.3, alle W-zulässigen Startzeitpunkte der Vorgänge eines Projekts. Das Zeitplanungsverfahren in Algorithmus 4.4 basiert auf einer schrittweisen Verkleinerung der Zeitpunkte innerhalb einer Startzeitbeschränkung W_i, beginnend mit dem größten Zeitpunkt $t = \max W_i$. Wie auch in Algorithmus 4.3 werden für jeden Vorgang $i \in V$ solange W-unzulässige Startzeitpunkte aus W_i entfernt und Anpassungen des Zeitpunkts t vorgenommen, bis schließlich kein W-zulässiger Startzeitpunkt mehr durch eine Verkleinerung von t erreicht werden kann. Die Bestimmung von t' für die Anpassung des Zeitpunkts t sowie die Berechnung der indirekten zeitlichen Höchstabstände in jeder Iteration, die in der Liste $[\widehat{d}_{ij}(W, t)]$ abgespeichert werden, basieren auf Satz 4.5.

Satz 4.5. *Gegeben seien* $LS' = \max \widehat{\mathcal{S}}_T(W, i, t)$ *und* $LS'' = \max \widehat{\mathcal{S}}_T(W, i, \tau)$*. Dann gilt für jeden Zeitpunkt* $\tau \in W_i$ *mit* $\max(\max_{j \in V \setminus \{i\}}(s_j(LS'_j) + d_{ji}), d_{0i}) \leq \tau \leq LS'_i$ *die Bedingung* $LS''_h = \min(LS'_h, \tau - d_{hi})$ *für jeden Vorgang* $h \in V$*.*

Beweis. Analog zum Beweis von Satz 4.3. □

Nach der Ausführung von Algorithmus 4.4 werden die Höchstabstandsmatrix $\widehat{D}(W) := ([\widehat{d}_{ij}(W, t)])_{i,j \in V}$ sowie die Startzeitbeschränkung W, die ausschließlich alle W-zulässigen Startzeitpunkte der anfänglichen Startzeitbeschränkung enthält, ausgegeben. Der indirekte zeitliche Höchstabstand $\widehat{d}_{ij}(W, \tau)$ für jeden W-zulässigen Startzeitpunkt τ eines Vorgangs $i \in V$ kann durch

$$\widehat{d}_{ij}(W, \tau) := \widehat{d}_{ij}(W, \tau') + \operatorname{sgn}\left(\widehat{d}_{ij}(W, \tau'') - \widehat{d}_{ij}(W, \tau')\right) \cdot (\tau - \tau')$$

mit $\tau' := \max\{\sigma \in \Psi \mid \sigma \leq \tau\}$, $\tau'' := \min\{\sigma \in \Psi \mid \sigma \geq \tau\}$ und Ψ als Menge aller abgespeicherten Zeitpunkte in $[\widehat{d}_{ij}(W, t)]$ berechnet werden.

Satz 4.6. *Algorithmus 4.4 bestimmt für jeden Vorgang* $i \in V$ *alle* W*-zulässigen Startzeitpunkte und den indirekten zeitlichen Höchstabstand* $\widehat{d}_{ij}(W, t)$ *zu jedem Vorgang* $j \in V \setminus \{i\}$ *für eine Teilmenge aller Zeitpunkte* $t \in \mathcal{H}$*, durch die der zeitliche Höchstabstand* $\widehat{d}_{ij}(W, \tau)$ *für jeden* W*-zulässigen Startzeitpunkt* $\tau \in W_i$ *bestimmt werden kann, mit einer Zeitkomplexität von* $\mathcal{O}(|V|^2|E|(\mathcal{B}+1))$*.*

Beweis. Analog zum Beweis von Satz 4.4 mit entsprechender Anpassung der Lemmata 4.2, 4.3 und 4.4. □

Es ist zu beachten, dass in Kreter (2016, Abschnitt 5.1) und Kreter et al. (2016) unterschiedliche auf dem Floyd-Warshall-Algorithmus basierende Zeitplanungsverfahren zur Projektplanung mit allgemeinen Zeitbeziehungen und Kalendern vorgestellt wurden, die ebenfalls zur Bestimmung aller W-zulässigen Startzeitpunkte sowie der indirekten zeitlichen Mindestabstände zwischen allen Vorgangspaaren verwendet werden können. Für die Verfahren konnten Zeitkomplexitäten von $\mathcal{O}(\max(|V|^3\bar{d}^3, |V|^4\bar{d}^2))$, $\mathcal{O}(|V|^4\bar{d}^2)$ und $\mathcal{O}(\max(|V|^7(\mathcal{B}+1)^3, |V|^8(\mathcal{B}+1)^2))$ in Bezug auf die Notation der vorliegenden Dissertation gezeigt werden, während die Zeitplanungsverfahren, die in diesem Abschnitt vorgestellt wurden, jeweils eine Zeitkomplexität von $\mathcal{O}(|V|^2|E|(\mathcal{B}+1))$ aufweisen. Darüber hinaus stellt Algorithmus 4.4 das einzige dem Autor dieser Dissertation bekannte Verfahren zur Bestimmung der indirekten zeitlichen Höchstabstände zwischen den Vorgangspaaren für ein Projektplanungsproblem mit Startzeitbeschränkungen dar.

4.2 Konsistenztests

Konsistenztests werden zur Einschränkung eines betrachteten Suchraums durch die Herleitung impliziter Restriktionen eingesetzt, wobei der verkleinerte Suchraum weiterhin alle zulässigen Lösungen enthält. Die impliziten Restriktionen werden aus den Wertebereichen der Entscheidungsvariablen und den durch die explizit gegebenen Restriktionen bestehenden Zusammenhänge abgeleitet. Allgemein kann jeder Konsistenztest durch eine Bedingung und eine Restriktion beschrieben werden, wobei die Restriktion dem betrachteten Suchraum nur dann hinzugefügt wird, falls die zugehörige Bedingung erfüllt ist (vgl. Schwindt, 2005, Abschnitt 1.2.4). Zur Einschränkung eines Suchraums werden in der Regel verschiedene Konsistenztests solange ausgeführt, bis entweder durch keinen der betrachteten Konsistenztests weitere Restriktionen abgeleitet werden können oder gezeigt wird, dass keine zulässige Lösung im Suchraum existiert. Dieser Zustand wird auch als Fixpunkt der Konsistenztests bezeichnet.

Im Folgenden werden verschiedene Konsistenztests zum RCPSP/max-π vorgestellt, die in jedem Knoten der Enumerationsverfahren, die in Kapitel 5 beschrieben werden, verwendet werden können. Alle Konsistenztests in diesem Abschnitt können entsprechend zur Einschränkung des Suchraums $\mathcal{S}_T(W)$ eines beliebigen Enumerationsknotens eingesetzt werden. Für jeden der nachfolgend betrachteten Konsistenztests kann die abgeleitete Restriktion direkt in eine Startzeitreduktion überführt werden, d. h., es wird mindestens ein Zeitpunkt aus der Startzeitbeschränkung W_i eines Vorgangs eliminiert, wenn die Bedingung des betrachteten Konsistenztests erfüllt ist. In Anlehnung an Dorndorf et al. (2000b) werden alle Konsistenztests in diesem Abschnitt unter dem Begriff Domain-Konsistenztests zusammengefasst, wobei jeder der Konsistenztests als eine Funktion γ aufgefasst werden kann, die einer Startzeitbeschränkung W eine Startzeitbeschränkung $W' = \gamma(W)$ mit $W'_i \subseteq W_i$ für alle $i \in V$ zuordnet. Im Fall der in diesem Abschnitt betrachteten Domain-Konsistenztests kann der Fixpunkt durch eine Startzeitbeschränkung W angegeben werden, für die $\gamma(W) = W$ für alle betrachteten Konsistenztests $\gamma \in \Gamma$ oder $W_i = \emptyset$ für mindestens einen Vorgang $i \in V$ gilt.

Die ersten Konsistenztests, die in diesem Abschnitt vorgestellt werden, basieren auf den Zeitbeziehungen $S_j \geq S_i + \delta_{ij}$ für alle Vorgangspaare $(i, j) \in E$ und können daher auch unabhängig vom betrachteten Ressourcentyp für andere Projektplanungsprobleme als für das RCPSP/max-π eingesetzt werden. Zunächst werden sogenannte Vorrang-Konsistenztests betrachtet, die bereits für Projektplanungsprobleme mit Vorrangbeziehungen (vgl. z. B. Dorndorf et al., 2000a; Alvarez-Valdes et al., 2006) und mit allgemeinen Zeitbeziehungen (vgl. z. B. Dorndorf et al., 2000c) angewendet und untersucht wurden.

Im Anschluss daran wird erläutert, wie der Fixpunkt dieser Konsistenztests durch die Zeitplanungsverfahren in Abschnitt 4.1.1 bestimmt werden kann. In jeder Iteration der Vorrang-Konsistenztests wird für alle Zeitbeziehungen $(i,j) \in E$ überprüft, ob die Zeitpunkte $t \in W_j$ die Bedingung $t \geq \min W_i + \delta_{ij}$ und die Zeitpunkte $t \in W_i$ die Bedingung $t \leq \max W_j - \delta_{ij}$ erfüllen. Es wird dabei ausgenutzt, dass $\min W_i + \delta_{ij}$ eine untere Schranke für den Startzeitpunkt S_j und $\max W_j - \delta_{ij}$ eine obere Schranke für den Startzeitpunkt S_i darstellt. Die Konsistenztests können durch die folgenden Bedingungen und Startzeitreduktionen für jede Zeitbeziehung $(i,j) \in E$ dargestellt werden:

$$\begin{aligned} \min W_j < \min W_i + \delta_{ij} \quad &\Rightarrow \quad W_j := W_j \setminus [0, \min W_i + \delta_{ij}[\\ \max W_i > \max W_j - \delta_{ij} \quad &\Rightarrow \quad W_i := W_i \setminus]\max W_j - \delta_{ij}, \infty[\end{aligned}$$

Als Nächstes wird der Zusammenhang der Vorrang-Konsistenztests zu den Zeitplanungsverfahren in Abschnitt 4.1.1 erläutert. Sei W die Startzeitbeschränkung vor der Durchführung der Vorrang-Konsistenztests, W' die Startzeitbeschränkung nach einer beliebigen Iteration und W'' ein Fixpunkt. Dann kann in Anlehnung an den Beweis von Satz 4.1 gezeigt werden, dass die Zusammenhänge $(\min W_i'')_{i \in V} = ES(W)$ und $(\max W_i'')_{i \in V} = LS(W)$ für $\mathcal{S}_T(W) \neq \emptyset$ gelten. Zunächst sind die Bedingungen $\min W_i \leq ES_i(W)$ für alle Vorgänge $i \in V$ erfüllt. Da daraus $\min W_i' + \delta_{ij} \leq ES_j(W)$ in jeder Iteration folgt und der Fixpunkt für $\mathcal{S}_T(W) \neq \emptyset$ erst vorliegt, wenn die Bedingungen $\min W_j \geq \min W_i + \delta_{ij}$ für alle $(i,j) \in E$ erfüllt sind, gilt $(\min W_i'')_{i \in V} = ES(W)$. Analog kann $(\max W_i'')_{i \in V} = LS(W)$ für $\mathcal{S}_T(W) \neq \emptyset$ abgeleitet werden. Der Fixpunkt W'' der Vorrang-Konsistenztests kann somit durch die Algorithmen 4.1 und 4.2 mit einer Zeitkomplexität von $\mathcal{O}(|V||E|(\mathcal{B}+1))$ bestimmt werden. Für den Fall, dass ein W-zulässiger Schedule für die Startzeitbeschränkung W existiert, wird $W_i'' := W_i \cap [ES_i(W), LS_i(W)]$ für alle Vorgänge $i \in V$ mit einer Zeitkomplexität $\mathcal{O}(|V| + \mathcal{B})$ gesetzt. Die Vorrang-Konsistenztests werden im weiteren Verlauf dieser Arbeit durch den Temporal-Bound (TB) Konsistenztest zusammengefasst, der durch die Bedingung und die zugehörige Startzeitreduktion

$$t \notin [ES_i(W), LS_i(W)] \quad \Rightarrow \quad W_i := W_i \setminus \{t\}$$

für jeden Vorgang $i \in V$ und jeden Startzeitpunkt $t \in W_i$ beschrieben wird.

Der nächste Konsistenztest basiert ebenfalls auf den Zeitbeziehungen $S_j \geq S_i + \delta_{ij}$ für alle Vorgangspaare $(i,j) \in E$, wobei im Gegensatz zum TB-Konsistenztest nicht nur die Startzeitpunkte $t \in W_i$ mit $t < ES_i(W)$ oder $t > LS_i(W)$ aus W_i entfernt werden, sondern auch alle weiteren nicht W-zulässigen Startzeitpunkte. Der Temporal (T)

Konsistenztest wird durch

$$\nexists S \in \mathcal{S}_T(W) : S_i = t \quad \Rightarrow \quad W_i := W_i \setminus \{t\}$$

beschrieben, wobei eine Iteration dieses Konsistenztests die Bedingung für jeden Vorgang $i \in V$ und alle Zeitpunkte $t \in W_i$ überprüft. Wie in Abschnitt 4.1.2 gezeigt wurde, kann der Fixpunkt des T-Konsistenztests durch Algorithmus 4.3 (oder Algorithmus 4.4) mit einer Zeitkomplexität von $\mathcal{O}(|V|^2|E|(\mathcal{B}+1))$ bestimmt werden. Aus der Definition der W-zulässigen Startzeitpunkte folgt direkt, dass der T-Konsistenztest den TB-Konsistenztest in dem Sinne dominiert, dass $W_i^{\mathrm{T}} \subseteq W_i^{\mathrm{TB}}$ für alle $i \in V$ mit W^{T} als Fixpunkt des T-Konsistenztests und W^{TB} als Fixpunkt des TB-Konsistenztests gilt.

Im Folgenden werden Konsistenztests für das RCPSP/max-π vorgestellt, die auf den Ressourcenrestriktionen $r_k^c(S) \leq R_k$ für alle partiell erneuerbaren Ressourcen $k \in \mathcal{R}$ basieren und zum Teil die Zeitrestriktionen für weitere Einschränkungen mit einbeziehen. Alle diese Konsistenztests haben gemeinsam, dass für jeden Startzeitpunkt eines Vorgangs überprüft wird, ob die Ressourceninanspruchnahme zusammen mit den Mindestressourceninanspruchnahmen aller anderen Vorgänge des Projekts die Kapazität mindestens einer Ressource übersteigt. Die Mindestressourceninanspruchnahme eines Vorgangs wird dabei für jeden der nachfolgenden Konsistenztests unterschiedlich berechnet. Zunächst wird der Resource-Bound (RB) Konsistenztest beschrieben, der ausschließlich die Ressourcenrestriktionen berücksichtigt, sodass die Zeitbeziehungen zwischen allen Vorgängen des Projekts vernachlässigt werden. Entsprechend beeinflusst der betrachtete Startzeitpunkt $t \in W_i$ eines Vorgangs $i \in V$ nicht die möglichen Startzeitpunkte aller anderen Vorgänge $j \in V \setminus \{i\}$ des Projekts, sodass die Mindestinanspruchnahme einer Ressource $k \in \mathcal{R}$ für Vorgang j durch

$$r_{jk}^{c,min}(W) := \min\{r_{jk}^c(\tau) \mid \tau \in W_j\}$$

gegeben ist. Für jede Ressource $k \in \mathcal{R}$ wird die Mindestinanspruchnahme $r_{jk}^{c,min}(W)$ für jeden Vorgang $j \in V_k := \{i \in V \mid r_{ik}^d > 0\}$ bestimmt. Der RB-Konsistenztest wird durch die Bedingung und die zugehörige Startzeitreduktion

$$\exists k \in \mathcal{R}_i : r_{ik}^c(t) + \sum_{j \in V_k \setminus \{i\}} r_{jk}^{c,min}(W) > R_k \quad \Rightarrow \quad W_i := W_i \setminus \{t\}$$

mit $\mathcal{R}_i := \{k \in \mathcal{R} \mid r_{ik}^d > 0\}$ beschrieben. Innerhalb einer Iteration des RB-Konsistenztests wird die Bedingung für jeden Vorgang $i \in V \setminus \{0, n+1\}$ und alle Zeitpunkte $t \in W_i$ überprüft. Eine Iteration des RB-Konsistenztests kann mit einer Zeitkomplexität von

$\mathcal{O}(|V|\mathcal{I} + |\mathcal{R}|\mathcal{B})$ mit $\mathcal{I}$ als Anzahl der Belegungsintervalle über die Periodenmengen Π_k aller Ressourcen $k \in \mathcal{R}$ bestimmt werden. Eine nichtleere Menge $I := \{s, s+1, \ldots, e\} \subset \mathbb{Z}$ wird dabei genau dann als Belegungsintervall von Π_k mit den Start- und Endperioden s und e bezeichnet, wenn die Bedingungen $I \subseteq \Pi_k$ und $s-1, e+1 \notin \Pi_k$ erfüllt sind. Die Durchführung einer Iteration des RB-Konsistenztests mit einer Zeitkomplexität von $\mathcal{O}(|V|\mathcal{I} + |\mathcal{R}|\mathcal{B})$ wird dadurch erreicht, dass die Ressourcenbelegungen $r^u_{ik}(t)$ für jeden Vorgang $i \in V$ und jede Ressource $k \in \mathcal{R}_i$ nur für eine Teilmenge aller Zeitpunkte des Planungshorizonts $t \in \mathcal{H}$ in einer nach nichtfallenden Werten von t sortierten Liste $[r^u_{ik}(t)]$ abgespeichert werden. Die Zeitpunkte $\tau \in \mathcal{H}$ und die zugehörigen Ressourcenbelegungen, die in der Liste $[r^u_{ik}(t)]$ abgespeichert werden müssen, können aus den folgenden Zusammenhängen abgeleitet werden:

$$\begin{aligned}
\tau+1 \in \Pi_k \wedge \tau+p_i+1 \in \Pi_k &\Rightarrow r^u_{ik}(\tau+1) = r^u_{ik}(\tau) \\
\tau+1 \in \Pi_k \wedge \tau+p_i+1 \notin \Pi_k &\Rightarrow r^u_{ik}(\tau+1) = r^u_{ik}(\tau) - 1 \\
\tau+1 \notin \Pi_k \wedge \tau+p_i+1 \in \Pi_k &\Rightarrow r^u_{ik}(\tau+1) = r^u_{ik}(\tau) + 1 \\
\tau+1 \notin \Pi_k \wedge \tau+p_i+1 \notin \Pi_k &\Rightarrow r^u_{ik}(\tau+1) = r^u_{ik}(\tau)
\end{aligned} \tag{4.2}$$

Aus den Beziehungen zwischen den Zeitpunkten τ und $\tau+1$ folgt, dass die Bestimmung der Ressourcenbelegungen für alle Zeitpunkte $\tau \in \Psi := \{\tau' \in \mathcal{H} \mid \tau' \in \mathcal{U}_k \vee \tau' + p_i \in \mathcal{U}_k\}$ mit

$$\mathcal{U}_k := \{\sigma \mid \sigma \notin \Pi_k \wedge \sigma+1 \in \Pi_k\} \cup \{\sigma \mid \sigma \in \Pi_k \wedge \sigma+1 \notin \Pi_k\} \cup \{0, \bar{d}\}$$

ausreichend ist, um die Ressourcenbelegungen für alle Zeitpunkte $\tau \in \mathcal{H}$ bestimmen zu können. Es ist dabei zu beachten, dass Ψ der Menge der Stützstellen einer Funktion entspricht, die sich aus der linear-interpolierenden stetigen Fortsetzung der Ressourcenbelegungen $r^u_{ik}(\tau)$ über alle Zeitpunkte $\tau \in \mathcal{H}$ ergeben würde. Analog zu den Listen $[\widetilde{d}_{ij}(W,t)]$ und $[\widehat{d}_{ij}(W,t)]$ kann die Ressourcenbelegung $r^u_{ik}(\tau)$ für jeden Zeitpunkt $\tau \in \mathcal{H}$ durch

$$r^u_{ik}(\tau) := r^u_{ik}(\tau') + \operatorname{sgn}\left(r^u_{ik}(\tau'') - r^u_{ik}(\tau')\right) \cdot (\tau - \tau') \tag{4.3}$$

mit $\tau' := \max\{\sigma \in \Psi \mid \sigma \leq \tau\}$, $\tau'' := \min\{\sigma \in \Psi \mid \sigma \geq \tau\}$ und Ψ als Menge aller abgespeicherten Zeitpunkte in $[r^u_{ik}(t)]$ berechnet werden.

Um die Bestimmung einer Liste $[r^u_{ik}(t)]$ zu veranschaulichen, wird das Beispiel in Abbildung 4.3 betrachtet. In der Abbildung wird der Verlauf der Ressourcenbelegung einer Ressource $k \in \mathcal{R}_i$ durch einen Vorgang $i \in V$ über alle Zeitpunkte des Planungshorizonts $t \in \mathcal{H}$ mit $\Pi_k = \{3, \ldots, 7, 12\}$ und $p_i = 3$ gezeigt. Die Ressourcenbelegung $r^u_{ik}(t)$ für den Zeitpunkt $t = 0$ und die Periodenmenge Π_k werden jeweils durch schraffierte Flächen verdeutlicht. Für das Beispiel ergibt sich $\mathcal{U}_k = \{\sigma \mid \sigma \notin \Pi_k \wedge \sigma+1 \in \Pi_k\} \cup \{\sigma \mid \sigma \in$

$\Pi_k \wedge \sigma + 1 \notin \Pi_k\} \cup \{0, \bar{d}\} = \{2, 11\} \cup \{7, 12\} \cup \{0, 15\} = \{0, 2, 7, 11, 12, 15\}$, sodass $\Psi = \{\tau' \in \mathcal{H} \mid \tau' \in \mathcal{U}_k\} \cup \{\tau' \in \mathcal{H} \mid \tau' + p_i \in \mathcal{U}_k\} = \{0, 2, 7, 11, 12, 15\} \cup \{4, 8, 9, 12\} = \{0, 2, 4, 7, 8, 9, 11, 12, 15\}$ direkt abgeleitet werden kann. Die Zeitpunkte $\tau \in \Psi$, für die es ausreicht, die Ressourcenbelegungen in der Liste $[r_{ik}^u(t)]$ abzuspeichern, sind in Abbildung 4.3 durch Quadrate markiert. Die Verbindung der Quadrate durch gestrichelte Linien soll die Bestimmung der Ressourcenbelegung für jeden Zeitpunkt $\tau \in \mathcal{H}$ durch die Berechnungsvorschrift (4.3) verdeutlichen.

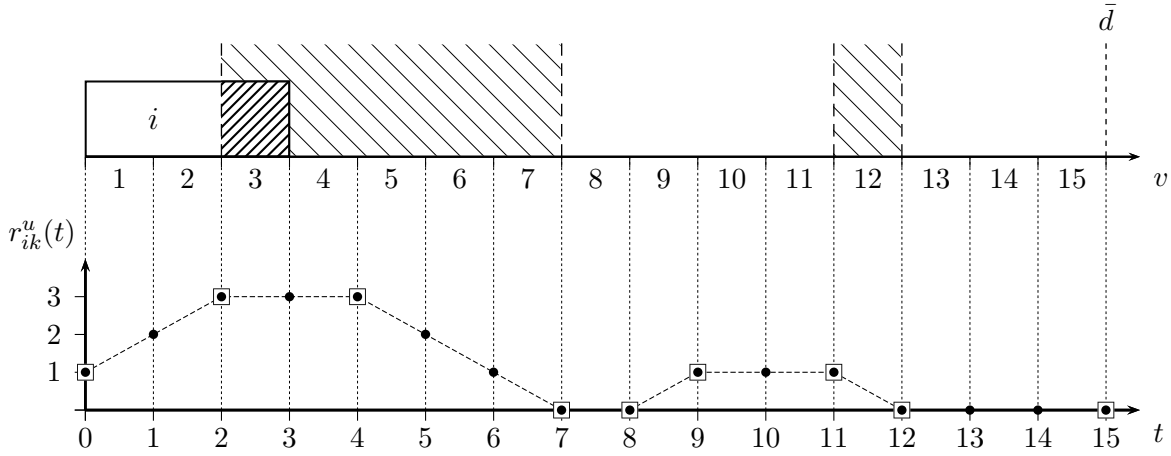

Abbildung 4.3: Verlauf der Ressourcenbelegung über den Planungshorizont

Aus den Beschreibungen zur Erzeugung einer Liste $[r_{ik}^u(t)]$ folgt schließlich, dass die Anzahl der abgespeicherten Zeitpunkte und der zugehörigen Ressourcenbelegungen in einer Liste $[r_{ik}^u(t)]$ durch $\mathcal{O}(\mathcal{I}_k)$ mit $\mathcal{I}_k$ als Anzahl der Belegungsintervalle in Π_k polynomiell beschränkt ist. Die minimale Ressourcenbelegung

$$r_{ik}^{u,min}(W) := \min\{r_{ik}^u(\tau) \mid \tau \in W_i\}$$

kann entsprechend mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{I}_k + \mathcal{B}_i)$ berechnet werden, woraus für die Bestimmung der Mindestinanspruchnahmen $r_{ik}^{c,min}(W) = r_{ik}^{u,min}(W)\, r_{ik}^d$ für alle Vorgänge $i \in V_k$ und Ressourcen $k \in \mathcal{R}$ innerhalb einer Iteration des RB-Konsistenztests eine Zeitkomplexität von $\mathcal{O}(|V|\mathcal{I} + |\mathcal{R}|\mathcal{B})$ folgt. Da weiterhin alle inkonsistenten Startzeitpunkte $t \in W_i$ für jede Ressource $k \in \mathcal{R}_i$ mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{I}_k + \mathcal{B}_i)$ aus W_i entfernt werden können, ergibt sich eine Zeitkomplexität von $\mathcal{O}(|V|\mathcal{I} + |\mathcal{R}|\mathcal{B})$ für eine Iteration des RB-Konsistenztests. Für jede Instanz des RCPSP/max-π wird im weiteren Verlauf dieser Arbeit vorausgesetzt, dass zu Beginn für jeden Vorgang $i \in V$ und jede Ressource $k \in \mathcal{R}_i$ eine Liste $[r_{ik}^u(t)]$ erzeugt wird. Die Zeitkomplexität von $\mathcal{O}(|V|\mathcal{I})$ für die Bestimmung aller Listen ergibt sich direkt aus den Beziehungen (4.2) und der

Verwaltung aller Periodenmengen Π_k als Listen, die nur die Start- und Endperioden der Belegungsintervalle beinhalten.

Im Folgenden wird ein Konsistenztest, der in Alvarez-Valdes et al. (2006) für das Projektplanungsproblem RCPSP/π entwickelt wurde, um allgemeine Zeitbeziehungen zwischen den Vorgängen des Projekts erweitert. Das entsprechende Verfahren stellt eine Erweiterung des RB-Konsistenztests dar, wobei neben den Ressourcenrestriktionen zusätzlich auch die Zeitbeziehungen zwischen den Vorgängen des Projekts berücksichtigt werden. Es wird dabei ausgenutzt, dass für jeden zulässigen Schedule der Startzeitpunkt eines Vorgangs auf ein Zeitfenster eingegrenzt werden kann, falls der Startzeitpunkt eines anderen Vorgangs bereits festgelegt wurde. Zur Eingrenzung der Startzeitpunkte $\tau \in W_j$ eines Vorgangs $j \in V$ wird für den D-Konsistenztest die Distanzmatrix $D = (d_{ij})_{i,j \in V}$ betrachtet. Für jeden Startzeitpunkt $t \in W_i$ eines Vorgangs $i \in V$ wird dabei ein Startzeitfenster $[t + d_{ij}, t - d_{ji}]$ für jeden anderen Vorgang $j \in V \setminus \{i\}$ bestimmt. Der D-Konsistenztest wird für jeden D-zulässigen Startzeitpunkt $t \in W_i$ eines Vorgangs $i \in V$ ausgeführt, wobei ein Startzeitpunkt $t \in W_i$ genau dann als D-zulässig bezeichnet wird, wenn $[t+d_{ij}, t-d_{ji}] \cap W_j \neq \emptyset$ für alle $j \in V$ gilt. Die Mindestinanspruchnahme einer Ressource $k \in \mathcal{R}$ durch einen Vorgang $j \in V \setminus \{i\}$ über alle Zeitpunkte $\tau \in [t+d_{ij}, t-d_{ji}] \cap W_j$ für einen Startzeitpunkt $t \in W_i$ von Vorgang i ist durch

$$r_{ijkt}^{c,min}(W, D) := \min\{r_{jk}^{c}(\tau) \mid \tau \in W_j \cap [t + d_{ij}, t - d_{ji}]\}$$

gegeben. Entsprechend kann der D-Konsistenztest, der für jeden Vorgang $i \in V$ und alle D-zulässigen Startzeitpunkte in W_i ausgeführt wird, durch die Bedingung und die zugehörige Startzeitreduktion

$$\exists k \in \mathcal{R} : r_{ik}^{c}(t) + \sum_{j \in V_k \setminus \{i\}} r_{ijkt}^{c,min}(W, D) > R_k \quad \Rightarrow \quad W_i := W_i \setminus \{t\}$$

beschrieben werden. Algorithmus 4.5 skizziert den Verlauf einer Iteration des D-Konsistenztests.

Im Folgenden wird gezeigt, wie eine Iteration des D-Konsistenztests durch die Generierung von Listen in polynomieller Zeit in der Anzahl an Startzeitunterbrechungen der betrachteten Startzeitbeschränkung W ausgeführt werden kann. Im ersten Schritt wird zunächst die Bestimmung der Liste $[r_{ijkt}^{u,min}(W, D)]$ erläutert, die sich aus den Mindestbelegungen

$$r_{ijkt}^{u,min}(W, D) := \min\{r_{jk}^{u}(\tau) \mid \tau \in W_j \cap [t + d_{ij}, t - d_{ji}]\}$$

Algorithmus 4.5: D-Konsistenztest

Input: Startzeitbeschränkung W
Output: $\gamma^{\mathrm{D}}(W)$

```
1 forall i ∈ V do
2 |  forall k ∈ R do
3 |  |  forall j ∈ V_k \ {i} do
4 |  |  |  Bestimme Liste [r^{u,min}_{ijkt}(W, D)] (Algorithmus 4.6)
5 |  |  Erzeuge Liste [r^{c,min}_{ikt}(W, D)]
6 Aktualisiere W (gemäß ∃k ∈ R : r^{c,min}_{ikt}(W, D) > R_k  ⇒  W_i := W_i \ {t})
7 return W
```

einer Ressource $k \in \mathcal{R}$ für ein Vorgangspaar $(i, j) \in V \times V_k$, $i \neq j$ und eine Teilmenge aller Startzeitpunkte $t \in \mathcal{H}$ zusammensetzt. Die Liste $[r^{u,min}_{ijkt}(W, D)]$ kann analog zu den Berechnungsvorschriften (4.1) und (4.3) zur Bestimmung der Mindestressourcenbelegung $r^{u,min}_{ijk\tau}(W, D)$ für jeden D-zulässigen Startzeitpunkt $\tau \in W_i$ von Vorgang $i \in V$ verwendet werden. Das Vorgehen für die Generierung der Liste $[r^{u,min}_{ijkt}(W, D)]$ wird in Algorithmus 4.6 skizziert.

Zunächst wird in Zeile 1 von Algorithmus 4.6 überprüft, ob mindestens ein Startzeitpunkt $t \in W_i$ des Vorgangs $i \in V$ die Menge der Startzeitpunkte W_j von Vorgang $j \in V_k \setminus \{i\}$ einschränkt, d. h., ob $W^e_{ij}(t) \subset W_j$ mit $W^e_{ij}(t) := [t + d_{ij}, t - d_{ji}] \cap W_j$ gilt. Für den Fall, dass die Bedingung $W^e_{ij}(t) = W_j$ für jeden Zeitpunkt $t \in W_i$ erfüllt ist, wird die Mindestressourcenbelegung $r^{u,min}_{jk}(W)$ sowohl für den Zeitpunkt $t = \min W_i$ als auch für den Zeitpunkt $t = \max W_i$ in der Liste $[r^{u,min}_{ijkt}(W, D)]$ abgespeichert (Zeile 30). Falls dagegen $W^e_{ij}(t) \subset W_j$ für mindestens einen Zeitpunkt $t \in W_i$ gilt, werden alle Startzeitpunkte in der Startzeitbeschränkung W_i beginnend mit dem kleinsten Zeitpunkt $t = \min W_i$ durchlaufen. Zur Vereinfachung der Darstellung werden im Folgenden $\min \emptyset := \infty$ und $\max \emptyset := -\infty$ vorausgesetzt. Vor der ersten Iteration werden zunächst die Mindestressourcenbelegung $r^u_{min} := \min\{r^u_{jk}(\tau) \mid \tau \in W^e_{ij}(t)\}$ und der späteste Zeitpunkt $\tau \in W^e_{ij}(t)$ mit $r^u_{jk}(\tau) = r^u_{min}$ durch $\tau^{min} := \max\{\tau \in W^e_{ij}(t) \mid r^u_{jk}(\tau) = r^u_{min}\}$ für den Zeitpunkt $t = \min W_i$ berechnet. Im Anschluss daran wird r^u_{min} für den Zeitpunkt $t = \min W_i$ in der Liste $[r^{u,min}_{ijkt}(W, D)]$ abgespeichert sowie die Binärvariable *shiftOnly* := *false* gesetzt, die im Verlauf des Algorithmus angibt, ob für den aktuell betrachteten Zeitpunkt t die Mindestressourcenbelegung r^u_{min} bereits in der Liste $[r^{u,min}_{ijkt}(W, D)]$ abgespeichert wurde (*shiftOnly* = *false*) oder nicht (*shiftOnly* = *true*). Solange $t < \max W_i$ gilt, wird in jeder Iteration von Algorithmus 4.6 der späteste Zeitpunkt t^s ermittelt, der durch eine Erhöhung des Zeitpunkts t ohne Veränderung von r^u_{min} erreicht werden kann. Zur Bestimmung von t^s werden die Zeitpunkte $\tau' := \min\{\tau \in W_j \mid \tau > t - d_{ji} \wedge r^u_{jk}(\tau) < r^u_{min}\}$

Algorithmus 4.6: Bestimmung der Liste $[r_{ijkt}^{u,min}(W, D)]$

Input: Startzeitbeschränkung W, Distanzmatrix D
Output: $[r_{ijkt}^{u,min}(W, D)]$

if $\min W_i - d_{ji} < \max W_j \ \vee \ \max W_i + d_{ij} > \min W_j$ **then**
  $t := \min W_i, \qquad r_{min}^u := \min\{r_{jk}^u(\tau) \mid \tau \in W_{ij}^e(t)\}$
  $\tau^{min} := \max\{\tau \in W_{ij}^e(t) \mid r_{jk}^u(\tau) = r_{min}^u\}$
  $r_{ijkt}^{u,min} := r_{min}^u, \qquad shiftOnly := false$
  while $t < \max W_i$ **do**
    $\tau' := \min\{\tau \in W_j \mid \tau > t - d_{ji} \wedge r_{jk}^u(\tau) < r_{min}^u\}$
    $\tau'' := \min\{\tau \in W_j \mid \tau > \tau^{min} \wedge r_{jk}^u(\tau) > r_{min}^u\}$
    $t' := t'' := \infty$
    if $\tau' < \infty$ **then** $t' := \tau' - 1 + d_{ji}$
    if $\tau'' < \infty$ **then** $t'' := \max\{\tau \in W_j \mid \tau < \tau''\} - d_{ij}$
    $t^s := \min(t', t'')$
    if $t^s < \max W_i$ **then**
      if $t^s = t'$ **then**
        Algorithmus 4.7
      else
        $r_{min}^{u,+} := \min\{r_{jk}^u(\tau) \mid \tau \in W_j \cap [t^s + 1 + d_{ij}, t^s + 1 - d_{ji}]\}$
        if $r_{min}^{u,+} = r_{min}^u$ **then**
          $t := t^s + 1, \qquad \tau^{min} := \max\{\tau \in W_{ij}^e(t) \mid r_{jk}^u(\tau) = r_{min}^{u,+}\}$
          $shiftOnly := true$
        else
          if $\tau'' = s_j(\tau'')$ **then**
            Algorithmus 4.8
          else
            Algorithmus 4.9
    else
      $t := \max W_i, \qquad shiftOnly := true$
  if $shiftOnly = true$ **then**
    $r_{ijkt}^{u,min} := r_{min}^u$
else
  $r_{min}^u := \min\{r_{jk}^u(\tau) \mid \tau \in W_j\}, \qquad r_{ijk,\min W_i}^{u,min} := r_{ijk,\max W_i}^{u,min} := r_{min}^u$
return $[r_{ijkt}^{u,min}]$

und $\tau'' := \min\{\tau \in W_j \mid \tau > \tau^{min} \wedge r_{jk}^u(\tau) > r_{min}^u\}$ ermittelt. Der Zeitpunkt τ' gibt den nächstgrößeren Startzeitpunkt von Vorgang $j \in V$ mit einer geringeren Ressourcenbelegung als r_{min}^u mit $\tau \in W_j$ und $\tau > t - d_{ji}$ an, wohingegen τ'' den nächstgrößeren Startzeitpunkt von Vorgang j mit einer größeren Ressourcenbelegung als r_{min}^u mit $\tau \in W_j$

und $\tau > \tau^{min}$ beschreibt. Entsprechend kann der Startzeitpunkt t von Vorgang i bis zum Zeitpunkt $t' = \tau' - 1 + d_{ji}$ ohne Verringerung der Mindestressourcenbelegung r^u_{min} und bis zum Zeitpunkt $t'' = \max\{\tau \in W_j \mid \tau < \tau''\} - d_{ij}$ ohne Anstieg von r^u_{min} erhöht werden. Der späteste Zeitpunkt t^s, bis zu dem eine Erhöhung des Startzeitpunkts t von Vorgang i ohne Veränderung von r^u_{min} durchgeführt werden kann, ergibt sich somit durch $t^s := \min(t', t'')$.

Falls die Bedingungen $t^s < \max W_i$ und $t^s = t'$ erfüllt sind, woraus eine Abnahme von r^u_{min} für den Startzeitpunkt $t^s + 1$ folgt, wird Algorithmus 4.7 ausgeführt. Zunächst wird der Zeitpunkt t^s mit der aktuellen Mindestressourcenbelegung r^u_{min} in der Liste $[r^{u,min}_{ijkt}(W, D)]$ abgespeichert, falls $t^s > t$ gilt oder Zeitpunkt t mit r^u_{min} in der letzten Iteration nicht abgespeichert wurde ($shiftOnly = true$). Es werden zwei Fälle in Algorithmus 4.7 in Bezug auf den Zeitpunkt τ' unterschieden. Im ersten Fall entspricht τ' dem Startzeitpunkt $s_j(\tau')$ des Startzeitintervalls I in W_j, dem τ' zugeordnet ist, wodurch r^u_{min} durch eine Erhöhung von t^s um eine Zeiteinheit beliebig abnehmen kann. Für den zweiten Fall mit $\tau' \neq s_j(\tau')$ nimmt r^u_{min} durch die Erhöhung von t^s um jeweils eine Einheit genau um eine Einheit ab, solange $t^s - d_{ji} \leq \tau^{dec}$ mit $\tau^{dec} := \max\{\tau \in W_j \cap [\tau', e_j(\tau')] \mid r^u_{jk}(v) < r^u_{jk}(v-1)\ \forall v \in W_j : \tau' \leq v \leq \tau\}$ erfüllt ist. Für beide Fälle gilt, dass es ausreicht, den Zeitpunkt $\tau^{dec} + d_{ji}$ mit $r^u_{min} := r^u_{jk}(\tau^{dec})$ in der Liste $[r^{u,min}_{ijkt}(W, D)]$ abzuspeichern, um $r^{u,min}_{ijk\tau}(W, D)$ für alle Startzeitpunkte τ von Vorgang i mit $\tau' + d_{ji} \leq \tau \leq \tau^{dec} + d_{ji}$ analog zu den Berechnungsvorschriften (4.1) und (4.3) bestimmen zu können. Am Ende von Algorithmus 4.7 werden schließlich die Werte für t, r^u_{min} und τ^{min} aktualisiert, r^u_{min} für den Zeitpunkt $t = \tau^{dec} + d_{ji}$ in der Liste $[r^{u,min}_{ijkt}(W, D)]$ abgespeichert und $shiftOnly := false$ gesetzt.

Algorithmus 4.7: Verringerung der Mindestressourcenbelegung

if $t^s > t \vee shiftOnly = true$ **then**
  $t := t^s, \qquad r^{u,min}_{ijkt} := r^u_{min}$
if $\tau' = s_j(\tau')$ **then**
  $\tau^{dec} := \tau'$
else
  $\tau^{dec} := \max\{\tau \in W_j \cap [\tau', e_j(\tau')] \mid r^u_{jk}(v) < r^u_{jk}(v-1)\ \forall v \in W_j : \tau' \leq v \leq \tau\}$
$t := \tau^{dec} + d_{ji}, \qquad r^u_{min} := r^u_{jk}(\tau^{dec}), \qquad \tau^{min} := \tau^{dec}$
$r^{u,min}_{ijkt} := r^u_{min}, \qquad shiftOnly := false$

Für den Fall, dass in einer Iteration von Algorithmus 4.6 die Bedingungen $t^s < \max W_i$ und $t^s \neq t'$ erfüllt sind ($t^s = t''$), wird zunächst überprüft, ob eine Erhöhung des Startzeitpunkts t^s von Vorgang i um eine Zeiteinheit zu einem Anstieg von r^u_{min} führt. Dafür

wird $r^{u,+}_{min} := \min\{r^u_{jk}(\tau) \mid \tau \in W_j \cap [t^s + 1 + d_{ij}, t^s + 1 - d_{ji}]\}$ bestimmt. Falls $r^{u,+}_{min} = r^u_{min}$ gilt, wird $t := t^s + 1$ gesetzt und der Zeitpunkt τ^{min} aktualisiert. Da der Startzeitpunkt von Vorgang i erhöht wird, ohne den aktualisierten Zeitpunkt $t := t^s + 1$ mit der zugehörigen Mindestressourcenbelegung r^u_{min} in der Liste $[r^{u,min}_{ijkt}(W, D)]$ abzuspeichern, wird *shiftOnly* := *true* gesetzt. Nimmt die Mindestressourcenbelegung durch die Erhöhung des Startzeitpunkts t^s um eine Zeiteinheit dagegen zu, d. h., gilt $r^{u,+}_{min} > r^u_{min}$, wird entweder Algorithmus 4.8 für den Fall $\tau'' = s_j(\tau'')$ oder Algorithmus 4.9 im Fall $\tau'' \neq s_j(\tau'')$ ausgeführt.

Algorithmus 4.8 erweitert zunächst wie Algorithmus 4.7 die Liste $[r^{u,min}_{ijkt}(W, D)]$ um den Zeitpunkt t^s mit der zugehörigen Mindestressourcenbelegung r^u_{min}, falls Zeitpunkt t^s noch nicht abgespeichert wurde. Für den Fall $r^{u,+}_{min} < \infty$ bzw. $W^e_{ij}(t^s + 1) \neq \emptyset$ werden die Werte für t, r^u_{min} und τ^{min} aktualisiert und die Mindestressourcenbelegung $r^{u,+}_{min}$ mit dem Zeitpunkt $t^s + 1$ der Liste $[r^{u,min}_{ijkt}(W, D)]$ hinzugefügt. Andernfalls wird eine Mindestressourcenbelegung $r^u_{min} = \infty$ für die Zeitpunkte t^s+1 und $\tau''+d_{ji}-1$ abgespeichert, um alle Zeitpunkte im Intervall $[t^s + 1, \tau'' + d_{ji} - 1]$ als D-unzulässig zu markieren. Anschließend werden alle Werte für $t = \tau'' + d_{ji}$ aktualisiert und $r^u_{min} = r^u_{jk}(\tau'')$ für den aktualisierten Zeitpunkt $t = \tau'' + d_{ji}$ in der Liste $[r^{u,min}_{ijkt}(W, D)]$ abgespeichert.

Algorithmus 4.8: Erhöhung der Mindestressourcenbelegung für $\tau'' = s_j(\tau'')$

```
1  if t^s > t ∨ shiftOnly = true then
2  |  t := t^s,        r^{u,min}_{ijkt} := r^u_{min}
3  if r^{u,+}_{min} < ∞ then
4  |  t := t^s + 1,    r^u_{min} := r^{u,+}_{min},    τ^{min} := max{τ ∈ W^e_{ij}(t) | r^u_{jk}(τ) = r^{u,+}_{min}}
5  |  r^{u,min}_{ijkt} := r^u_{min},    shiftOnly := false
6  else
7  |  t := t^s + 1,    r^{u,min}_{ijkt} := ∞
8  |  t := τ'' + d_{ji} − 1,    r^{u,min}_{ijkt} := ∞
9  |  t := τ'' + d_{ji},    r^u_{min} := r^u_{jk}(τ''),    τ^{min} := τ''
10 |  r^{u,min}_{ijkt} := r^u_{jk}(τ''),    shiftOnly := false
```

Vor der Ausführung von Algorithmus 4.9 folgt zunächst aus den Bedingungen $\tau'' \neq s_j(\tau'')$ und $r^{u,+}_{min} > r^u_{min}$, dass sich r^u_{min} durch die Erhöhung von t^s um eine Zeiteinheit genau um eine Einheit vergrößert. Algorithmus 4.9 bestimmt die maximale Zeitspanne, um die der Zeitpunkt t^s erhöht werden kann, sodass r^u_{min} durch jede Vergrößerung von t^s um eine Zeiteinheit genau um eine Einheit zunimmt. Nach der Speicherung von Zeitpunkt t^s und der zugehörigen Mindestressourcenbelegung r^u_{min} in der Liste $[r^{u,min}_{ijkt}(W, D)]$ für den Fall,

Algorithmus 4.9: Erhöhung der Mindestressourcenbelegung für $\tau'' \neq s_j(\tau'')$

if $t^s > t \vee \mathit{shiftOnly} = \mathit{true}$ **then**
  $t := t^s, \qquad r^{u,min}_{ijkt} := r^u_{min}$
$\tau^{inc} := \max\{\tau \in W_j \cap [\tau'', e_j(\tau'')] \mid r^u_{jk}(v) > r^u_{jk}(v-1)\ \forall v \in W_j : \tau'' \leq v \leq \tau\}$
if $\tau^{inc} < t - d_{ji}$ **then**
  $\bar{r}^u_{min} := \min\{r^u_{jk}(\tau) \mid \tau \in W_j \wedge \tau^{inc} \leq \tau \leq t - d_{ji}\}$
else
  $\bar{r}^u_{min} := r^u_{jk}(\tau^{inc})$
$\Delta^{max} := \bar{r}^u_{min} - r^u_{min}$
while *true* **do**
  $\tau^s := \min\{\tau \in W_j \mid \tau \geq \max(\tau^{inc}, t - d_{ji}) \wedge r^u_{jk}(\tau) < \bar{r}^u_{min}\}$
  $\Delta := \tau^s - 1 - (t - d_{ji})$
  if $\Delta \geq \Delta^{max}$ **then**
    $t := t + \Delta^{max}, \qquad r^u_{min} := r^u_{min} + \Delta^{max}$
    $\tau^{min} := \max\{\tau \in W^e_{ij}(t) \mid r^u_{jk}(\tau) = r^u_{min}\}$
    $r^{u,min}_{ijkt} := r^u_{min}, \qquad \mathit{shiftOnly} := \mathit{false}$
    break
  else
    if $\Delta > 0$ **then**
      $\Delta^{max} := \Delta^{max} - \Delta, \qquad t := t + \Delta, \qquad r^u_{min} := r^u_{min} + \Delta$
    if $\tau^s = s_j(\tau^s)$ **then**
      Algorithmus 4.10
    else
      Algorithmus 4.11

dass t^s dieser Liste noch nicht hinzugefügt wurde, wird der Zeitpunkt τ^{inc} bestimmt. Die Zeitspanne $\tau^{inc} - (t^s + d_{ij})$ entspricht der maximalen Erhöhung des Zeitpunkts t^s, für die gilt, dass sich bei einer Vergrößerung von t^s um jeweils eine Zeiteinheit r^u_{min} um eine Einheit erhöhen könnte. Da weiterhin die maximale Erhöhung von r^u_{min} um $\tau^{inc} - (t^s + d_{ij})$ Einheiten für den Fall $t^s - d_{ji} > \tau^{inc}$ durch die Mindestressourcenbelegung $\bar{r}^u_{min}$ im Zeitintervall $[\tau^{inc}, t^s - d_{ji}]$ eingeschränkt sein könnte, wird $\Delta^{max} := \bar{r}^u_{min} - r^u_{min}$ als maximal mögliche Erhöhung von r^u_{min} bzw. von Zeitpunkt t^s gesetzt. In jedem Iterationsschritt wird zunächst die Zeitspanne $\Delta := \tau^s - 1 - (t - d_{ji})$ ermittelt, um die der aktuelle Startzeitpunkt t von Vorgang i erhöht werden kann, sodass $\bar{r}^u_{min} := \min\{r^u_{jk}(\tau) \mid \tau \in W_j \wedge \tau^{inc} \leq \tau \leq \max(\tau^{inc}, t - d_{ji})\}$ nicht abnimmt. Entsprechend werden für $\Delta \geq \Delta^{max}$ die Werte $t := t + \Delta^{max}$ und $r^u_{min} := r^u_{min} + \Delta^{max}$ gesetzt, τ^{min} aktualisiert, die Liste

$[r_{ijkt}^{u,min}(W,D)]$ erweitert und Algorithmus 4.9 beendet (break). Für den Fall, dass $\bar{r}_{min}^{u}$ dagegen durch die Erhöhung von t um $\Delta + 1 \leq \Delta^{max}$ Zeiteinheiten abnimmt, werden zunächst die Werte Δ^{max}, t und r_{min}^{u} für $\Delta > 0$ angepasst. In Bezug auf den Zeitpunkt τ^s werden die Fälle $\tau^s = s_j(\tau^s)$ und $\tau^s \neq s_j(\tau^s)$ voneinander unterschieden, für die jeweils die Algorithmen 4.10 und 4.11 ausgeführt werden.

Algorithmus 4.10: Verringerung von $\bar{r}_{min}^{u}$ für $\tau^s = s_j(\tau^s)$

```
1  if r^u_jk(τ^s) < r^u_min then
2      r^{u,min}_ijkt := r^u_min
3      t := t + 1,        r^u_min := r^u_jk(τ^s),        τ^min := τ^s
4      r^{u,min}_ijkt := r^u_min,        shiftOnly := false
5      break
6  else if r^u_jk(τ^s) = r^u_min then
7      r^{u,min}_ijkt := r^u_min
8      t := t + 1,        τ^min := τ^s,        shiftOnly := true
9      break
10 else if r^u_jk(τ^s) = r^u_min + 1 then
11     t := t + 1,        r^u_min := r^u_min + 1,        τ^min := τ^s
12     r^{u,min}_ijkt := r^u_min,        shiftOnly := false
13     break
14 else
15     r̄^u_min := r^u_jk(τ^s),        Δ^max := r̄^u_min − (r^u_min + 1)
16     t := t + 1,        r^u_min := r^u_min + 1
```

Algorithmus 4.10 betrachtet den Fall $\tau^s = s_j(\tau^s)$. Es werden vier Fälle voneinander unterschieden, wobei die Algorithmen 4.9 und 4.10 beendet werden (break), falls $r_{jk}^{u}(\tau^s) \leq r_{min}^{u} + 1$ gilt. Für $r_{jk}^{u}(\tau^s) < r_{min}^{u}$ wird durch eine Erhöhung von t um eine Zeiteinheit eine geringere Mindestressourcenbelegung $r_{jk}^{u}(\tau^s)$ mit $\tau^{min} = \tau^s$ erreicht, sodass die Zeitpunkte t mit r_{min}^{u} und $t+1$ mit $r_{jk}^{u}(\tau^s)$ der Liste $[r_{ijkt}^{u,min}(W,D)]$ hinzugefügt werden. Gilt dagegen $r_{jk}^{u}(\tau^s) = r_{min}^{u}$, so verändert sich die Mindestressourcenbelegung r_{min}^{u} durch die Erhöhung des Zeitpunkts t um eine Zeiteinheit nicht, sodass der Zeitpunkt t mit r_{min}^{u} in der Liste $[r_{ijkt}^{u,min}(W,D)]$ abgespeichert wird und die Variablen durch $t := t+1$, $\tau^{min} := \tau^s$ und $shiftOnly := true$ aktualisiert werden. Im Fall $r_{jk}^{u}(\tau^s) = r_{min}^{u} + 1$ wird r_{min}^{u} durch eine Vergrößerung von Zeitpunkt t um eine Zeiteinheit weiterhin um eine Einheit erhöht. Entsprechend werden die Werte der Variablen für den Zeitpunkt $t+1$ angepasst und der Zeitpunkt $t := t+1$ mit $r_{min}^{u} := r_{min}^{u} + 1$ der Liste $[r_{ijkt}^{u,min}(W,D)]$ hinzugefügt. Im letzten

Fall $r^u_{jk}(\tau^s) > r^u_{min} + 1$ kann Zeitpunkt t um eine Zeiteinheit vergrößert werden, sodass r^u_{min} ebenfalls um eine Einheit erhöht wird und weiterhin die Bedingung $\bar{r}^u_{min} > r^u_{min}$ ($\Delta^{max} > 0$) gilt. Da r^u_{min} durch eine Erhöhung von t weiterhin zunehmen könnte, werden weitere Iterationen des Algorithmus 4.9 ausgeführt.

Algorithmus 4.11: Verringerung von $\bar{r}^u_{min}$ für $\tau^s \neq s_j(\tau^s)$

```
1  τ^dec := max{τ ∈ W_j ∩ [τ^s, e_j(τ^s)] | r^u_jk(v) < r^u_jk(v−1) ∀v ∈ W_j : τ^s ≤ v ≤ τ}
2  Δ^dec := min(Δ^max, τ^dec − (t − d_ji))
3  if r^u_min + Δ^dec ≥ r̄^u_min − Δ^dec then
4      Δ^h := ⌊(r̄^u_min − r^u_min)/2⌋
5      if r^u_min + Δ^h < r̄^u_min − Δ^h then
6          t := t + Δ^h,        r^u_min := r^u_min + Δ^h
7          r^{u,min}_ijkt := r^u_min
8          t := t + 1,          τ^min := t − d_ji,        shiftOnly := true
9      else
10         t := t + Δ^h,        r^u_min := r^u_min + Δ^h,        τ^min := t − d_ji
11         r^{u,min}_ijkt := r^u_min,        shiftOnly := false
12     break
13 else
14     r̄^u_min := r̄^u_min − Δ^dec
15     t := t + Δ^dec,        r^u_min := r^u_min + Δ^dec
16     Δ^max := Δ^max − 2 · Δ^dec
```

Aus $\tau^s \neq s_j(\tau^s)$ in Algorithmus 4.9 folgt zunächst, dass sich $\bar{r}^u_{min}$ jeweils um eine Einheit verringert, wenn sich Zeitpunkt t um eine Zeiteinheit erhöht, solange t maximal um $\Delta^{dec} := \min(\Delta^{max}, \tau^{dec} - (t - d_{ji}))$ mit $\tau^{dec} := \max\{\tau \in W_j \cap [\tau^s, e_j(\tau^s)] \mid r^u_{jk}(v) < r^u_{jk}(v-1)\ \forall v \in W_j : \tau^s \leq v \leq \tau\}$ Zeiteinheiten vergrößert wird. Da die Erhöhung von Zeitpunkt t um eine Zeiteinheit zu einer Zunahme von r^u_{min} um eine Einheit und zugleich zu einer Abnahme von $\bar{r}^u_{min}$ um eine Einheit führt, wird in Algorithmus 4.11 durch $r^u_{min} + \Delta^{dec} \geq \bar{r}^u_{min} - \Delta^{dec}$ zunächst überprüft, ob nach der Erhöhung von Zeitpunkt t um Δ^{dec} Zeiteinheiten $r^u_{min} \geq \bar{r}^u_{min}$ gilt. Falls die Bedingung $r^u_{min} + \Delta^{dec} \geq \bar{r}^u_{min} - \Delta^{dec}$ bzw. $\Delta^{dec} \geq \frac{\bar{r}^u_{min} - r^u_{min}}{2} = \frac{\Delta^{max}}{2}$ erfüllt ist, wird die Zeitspanne $\Delta^h := \left\lfloor \frac{\bar{r}^u_{min} - r^u_{min}}{2} \right\rfloor$ bestimmt. Es werden die beiden Fälle $\Delta^h < \frac{\Delta^{max}}{2}$ (Δ^{max} ist ungerade) und $\Delta^h = \frac{\Delta^{max}}{2}$ (Δ^{max} ist gerade) unterschieden. Für den Fall $\Delta^h < \frac{\Delta^{max}}{2}$ ist die Bedingung $r^u_{min} + \Delta^h < \bar{r}^u_{min} - \Delta^h$ erfüllt, wobei die Variablenwerte $t := t + \Delta^h$ und $r^u_{min} := r^u_{min} + \Delta^h$ gesetzt und in der Liste $[r^{u,min}_{ijkt}(W, D)]$ abgespeichert werden. Da durch eine weitere Erhöhung des Zeitpunkts t um eine Zeiteinheit die Mindestressourcenbelegung r^u_{min} unverändert bleibt, werden $t := t+1$,

$\tau^{min} := t - d_{ji}$ und *shiftOnly* := *true* gesetzt. Falls dagegen $\Delta^h = \frac{\Delta^{max}}{2}$ gilt, werden alle Variablenwerte durch $t := t + \Delta^h$, $r^u_{min} := r^u_{min} + \Delta^h$ und $\tau^{min} := t - d_{ji}$ aktualisiert und r^u_{min} für Zeitpunkt t der Liste $[r^{u,min}_{ijkt}(W, D)]$ hinzugefügt. In beiden Fällen werden die Algorithmen 4.9 und 4.11 beendet (break), da r^u_{min} durch eine weitere Vergrößerung von t um eine Zeiteinheit nicht weiter erhöht werden kann. Abschließend ist der Fall $r^u_{min} + \Delta^{dec} < \bar{r}^u_{min} - \Delta^{dec}$ bzw. $\Delta^{dec} < \frac{\bar{r}^u_{min} - r^u_{min}}{2} = \frac{\Delta^{max}}{2}$ zu betrachten. Zunächst werden die Variablen $r^u_{min} := r^u_{min} + \Delta^{dec}$, $t := t + \Delta^{dec}$ und $\bar{r}^u_{min} := \bar{r}^u_{min} - \Delta^{dec}$ aktualisiert, woraus schließlich $\Delta^{max} := \Delta^{max} - 2 \cdot \Delta^{dec} > 0$ ($\Delta^{dec} < \frac{\Delta^{max}}{2}$) für die nächste Iteration von Algorithmus 4.9 folgt.

Falls in einer Iteration von Algorithmus 4.6 die Bedingung $t^s \geq \max W_i$ erfüllt ist und somit die Mindestressourcenbelegung r^u_{min} bis zum Startzeitpunkt $\max W_i$ von Vorgang i konstant bleibt, werden die Variablen $t := \max W_i$ und *shiftOnly* := *true* gesetzt. Sobald $t \geq \max W_i$ nach einer Iteration von Algorithmus 4.6 gilt, wird abschließend der Zeitpunkt t mit der zugehörigen Mindestressourcenbelegung r^u_{min} der Liste $[r^{u,min}_{ijkt}(W, D)]$ hinzugefügt, falls r^u_{min} für den Zeitpunkt t noch nicht abgespeichert wurde. Die Ausgabe von Algorithmus 4.6 ist schließlich die Liste $[r^{u,min}_{ijkt}(W, D)]$.

Satz 4.7. *Algorithmus 4.6 bestimmt für ein Vorgangspaar $(i, j) \in V \times V_k$, $i \neq j$ die Mindestbelegung $r^{u,min}_{ijkt}(W, D)$ einer Ressource $k \in \mathcal{R}$ für eine Teilmenge aller Zeitpunkte $t \in \mathcal{H}$, durch die die Mindestressourcenbelegung $r^{u,min}_{ijk\tau}(W, D)$ für jeden D-zulässigen Startzeitpunkt $\tau \in W_i$ bestimmt werden kann, mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{B}_j^2 + \mathcal{I}_k^2)$.*

Beweis. Die Korrektheit von Algorithmus 4.6 kann aus den Beschreibungen zu den Algorithmen 4.6 bis 4.11 abgeleitet werden, sodass im Folgenden nur noch die Zeitkomplexität betrachtet wird. Im ersten Schritt wird zunächst gezeigt, dass Algorithmus 4.6 maximal $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ Iterationen durchläuft. Dafür wird angenommen, dass τ^{min}_λ und $\tau^{min}_{\lambda+1}$ den Werten der Variablen τ^{min} zu Beginn der Iterationen λ und $\lambda + 1$ von Algorithmus 4.6 entsprechen. Aus den Bedingungen $\tau' > \tau^{min}_\lambda$ und $\tau'' > \tau^{min}_\lambda$ folgt zunächst $\tau^{min}_{\lambda+1} > \tau^{min}_\lambda$, sodass τ^{min} über alle Iterationen von Algorithmus 4.6 streng monoton steigt. Weiterhin kann aus den Algorithmen 4.6 bis 4.11 abgeleitet werden, dass entweder die Bedingung $\tau^{min}_{\lambda+1} \geq e_j(\tau^{min}_\lambda)$ erfüllt ist oder $\tau^{min}_{\lambda+1}$ mindestens dem nächstgrößeren abgespeicherten Zeitpunkt $\tau > \tau^{min}_\lambda$ in der Liste $[r^u_{jk}(t)]$ entspricht. Da die Startzeitbeschränkung W_j aus $\mathcal{B}_j + 1$ Startzeitintervallen besteht und $[r^u_{jk}(t)]$ maximal $\mathcal{O}(\mathcal{I}_k)$ Einträge enthält, folgt schließlich eine maximale Anzahl an $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ Iterationen für Algorithmus 4.6. Im nächsten Schritt werden die Zeitkomplexitäten der Operationen der Algorithmen 4.6 bis 4.11 betrachtet. Für Algorithmus 4.6 ergibt sich zunächst eine Zeitkomplexität von $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ für die Bestimmung der Variablen r^u_{min} und τ^{min} in den Zeilen 2, 3 und 30. Weiterhin werden in jeder Iteration von Algorithmus 4.6 die Zeitpunkte τ' und τ'' sowie

die Variablen $r_{min}^{u,+}$ und τ^{min} mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ berechnet. Unter Berücksichtigung, dass bei den Berechnungen von τ' und $r_{min}^{u,+}$ der Endzeitpunkt $e_j(\tau')$, der nächstgrößere Zeitpunkt $\tau \geq \tau'$ in der Liste $[r_{jk}^u(t)]$ und der zugehörige Zeitpunkt τ^{min} zu $r_{min}^{u,+}$ abgespeichert werden, haben die Algorithmen 4.7 und 4.8 eine konstante Zeitkomplexität $\mathcal{O}(1)$. Da alle weiteren Operationen von Algorithmus 4.6 ebenfalls mit einer konstanten Zeitkomplexität ausgeführt werden, muss im Folgenden nur noch die Zeitkomplexität von Algorithmus 4.9 gezeigt werden. Durch die Speicherung des Endzeitpunkts $e_j(\tau'')$ und des nächstgrößeren Zeitpunkts $\tau \geq \tau''$ in der Liste $[r_{jk}^u(t)]$ bei der Bestimmung des Zeitpunkts τ'' in Algorithmus 4.6 werden τ^{inc} und $r_{jk}^u(\tau^{inc})$ mit einer konstanten Zeitkomplexität $\mathcal{O}(1)$ bestimmt. Weiterhin wird $\bar{r}_{min}^u$ für $\tau^{inc} < t^s - d_{ji}$ mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ berechnet, sodass alle Operationen außerhalb der Iterationsschleife von Algorithmus 4.9 mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ ausgeführt werden. Zur Bestimmung der maximalen Anzahl der Iterationsdurchläufe in Algorithmus 4.9 wird angenommen, dass τ_λ^s und $\tau_{\lambda+1}^s$ die Werte der aktualisierten Variablen τ^s zu Beginn der Iterationen λ und $\lambda+1$ darstellen. Da $\bar{r}_{min}^u$ über alle Iterationen von Algorithmus 4.9 streng monoton fällt, folgt zunächst $\tau_{\lambda+1}^s > \tau_\lambda^s$. Darüber hinaus ergibt sich aus den Algorithmen 4.10 und 4.11, dass mindestens eine der Bedingungen $\tau_{\lambda+1}^s > e_j(\tau_\lambda^s)$ oder $\tau_{\lambda+1}^s > \tau$ mit τ als dem nächstgrößeren abgespeicherten Zeitpunkt $\tau \geq \tau_\lambda^s$ in der Liste $[r_{jk}^u(t)]$ erfüllt ist. Entsprechend werden maximal $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ Iterationen in Algorithmus 4.9 ausgeführt, wobei τ^s über alle Iterationen mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ bestimmt werden kann, da τ^s streng monoton steigt. Aus der Zeitkomplexität für die Berechnung von τ^{min} in Zeile 14 mit $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ und dem konstanten Berechnungsaufwand für alle verbleibenden Operationen in Algorithmus 4.9 folgt abschließend eine Zeitkomplexität von $\mathcal{O}(\mathcal{B}_j^2 + \mathcal{I}_k^2)$ für die Bestimmung der Liste $[r_{ijkt}^{u,min}(W, D)]$. □

Nachdem in Algorithmus 4.5 die Listen $[r_{ijkt}^{u,min}(W, D)]$ aller Vorgänge $j \in V_k \backslash \{i\}$ für einen Vorgang $i \in V$ und eine Ressource $k \in \mathcal{R}$ bestimmt wurden, wird die Liste $[r_{ikt}^{c,min}(W, D)]$ erzeugt, die sich aus den Mindestressourceninanspruchnahmen

$$r_{ikt}^{c,min}(W, D) := r_{ik}^c(t) + \sum_{j \in V_k \backslash \{i\}} r_{ijkt}^{c,min}(W, D)$$

der Ressource k für eine Teilmenge aller Zeitpunkte $t \in \mathcal{H}$ zusammensetzt. Die Liste $[r_{ikt}^{c,min}(W, D)]$ ergibt sich direkt aus den Listen $[r_{ijkt}^{u,min}(W, D)]$ aller Vorgänge $j \in V_k \setminus \{i\}$ und $[r_{ik}^u(t)]$, indem die Mindestressourceninanspruchnahme $r_{ikt}^{c,min}(W, D)$ für jeden Zeitpunkt, der in mindestens einer der Listen abgespeichert ist, berechnet und der Liste $[r_{ikt}^{c,min}(W, D)]$ hinzugefügt wird. Die Liste $[r_{ikt}^{c,min}(W, D)]$ ist nach nichtfallenden Werten der Zeitpunkte t sortiert und kann analog zu den Berechnungsvorschriften (4.1) und (4.3) zur Bestimmung der Mindestressourceninanspruchnahme $r_{ik\tau}^{c,min}(W, D)$ für jeden D-

zulässigen Startzeitpunkt $\tau \in W_i$ von Vorgang i verwendet werden. Nach der Bestimmung aller Listen $[r^{u,min}_{ijkt}(W,D)]$ und $[r^{c,min}_{ikt}(W,D)]$ in Algorithmus 4.5 werden schließlich die Startzeitbeschränkungen aller Vorgänge des Projekts aktualisiert und die Startzeitbeschränkung $\gamma^{\mathrm{D}}(W)$ ausgegeben. Es ist dabei zu beachten, dass durch das Vorgehen zur Bestimmung der Listen $[r^{u,min}_{ijkt}(W,D)]$ im Allgemeinen nicht nur alle D-zulässigen Startzeitpunkte, die die Bedingung des D-Konsistenztests erfüllen, aus der Startzeitbeschränkung W entfernt werden, sondern auch D-unzulässige Startzeitpunkte eliminiert werden können.

Satz 4.8. *Algorithmus 4.5 hat eine Zeitkomplexität von* $\mathcal{O}(|V|^2(\mathcal{B}^2|\mathcal{R}| + |V|\mathcal{I}^2))$.

Beweis. Aus Satz 4.7 ergibt sich zunächst, dass die Listen $[r^{u,min}_{ijkt}(W,D)]$ aller Vorgänge $j \in V_k \setminus \{i\}$ für einen Vorgang $i \in V$ und eine Ressource $k \in \mathcal{R}$ mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{B}^2 + |V|\mathcal{I}_k^2)$ bestimmt werden können. Da in der Liste $[r^u_{ik}(t)]$ höchstens $\mathcal{O}(\mathcal{I}_k)$ Zeitpunkte abgespeichert sind und aus dem Beweis von Satz 4.7 hervorgeht, dass jede Liste $[r^{u,min}_{ijkt}(W,D)]$ maximal $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ Einträge hat, müssen für höchstens $\mathcal{O}(\mathcal{B} + |V|\mathcal{I}_k)$ Zeitpunkte die Mindestressourceninanspruchnahmen $r^{c,min}_{ikt}(W,D)$ für die Bestimmung der Liste $[r^{c,min}_{ikt}(W,D)]$ berechnet werden. Indem alle gespeicherten Zeitpunkte in den Listen $[r^u_{ik}(t)]$ und $[r^{u,min}_{ijkt}(W,D)]$ für alle Vorgänge $j \in V_k \setminus \{i\}$ nach nichtfallenden Werten durchlaufen werden, kann die Liste $[r^{c,min}_{ikt}(W,D)]$ mit einer Zeitkomplexität von $\mathcal{O}(|V|(\mathcal{B}+|V|\mathcal{I}_k))$ erzeugt werden. Die Bestimmung aller Listen $[r^{u,min}_{ijkt}(W,D)]$ und $[r^{c,min}_{ikt}(W,D)]$ für einen Vorgang $i \in V$ in Algorithmus 4.5 erfordert entsprechend $\mathcal{O}(|V|(\mathcal{B}^2|\mathcal{R}| + |V|\mathcal{I}^2))$ Operationen. Da die Aktualisierung der Startzeitbeschränkung W_i mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{B}|\mathcal{R}| + |V|\mathcal{I})$ ausgeführt werden kann, ergibt sich schließlich eine Zeitkomplexität von $\mathcal{O}(|V|^2(\mathcal{B}^2|\mathcal{R}|+|V|\mathcal{I}^2))$ für eine Iteration des D-Konsistenztests. □

Der letzte in diesem Abschnitt behandelte Konsistenztest stellt eine Erweiterung des D-Konsistenztests dar, indem die Mengen der Startzeitpunkte der Vorgänge $j \in V_k \setminus \{i\}$ durch die indirekten zeitlichen Mindest- und Höchstabstände $\widetilde{d}_{ij}(W,t)$ und $\widehat{d}_{ij}(W,t)$ für einen Startzeitpunkt $t \in W_i$ eines Vorgangs $i \in V$ eingeschränkt werden. Hierdurch werden im Gegensatz zum D-Konsistenztest die Startzeitunterbrechungen der Startzeitbeschränkung W berücksichtigt. Für den sogenannten W-Konsistenztest ist die Mindestinanspruchnahme einer Ressource $k \in \mathcal{R}$ durch einen Vorgang $j \in V_k \setminus \{i\}$ in Abhängigkeit vom Startzeitpunkt t eines Vorgangs $i \in V$ durch

$$r^{c,min}_{ijkt}(W,\widetilde{D},\widehat{D}) := \min\{r^c_{jk}(\tau) \mid \tau \in W_j \cap [t+\widetilde{d}_{ij}(W,t), t+\widehat{d}_{ij}(W,t)]\}$$

gegeben. Der W-Konsistenztest, der in einer Iteration für jeden Vorgang $i \in V$ über alle Zeitpunkte $t \in W_i$ ausgeführt wird, kann durch die folgende Bedingung und zugehörige Startzeitreduktion beschrieben werden:

$$\exists k \in \mathcal{R} : r_{ik}^{c}(t) + \sum_{j \in V_k \setminus \{i\}} r_{ijkt}^{c,min}(W, \widetilde{D}, \widehat{D}) > R_k \quad \Rightarrow \quad W_i := W_i \setminus \{t\}$$

Bislang wurden die Listen $[\widetilde{d}_{ij}(W,t)]$ und $[\widehat{d}_{ij}(W,t)]$ für die Berechnung der indirekten zeitlichen Mindest- und Höchstabstände für alle W-zulässigen Startzeitpunkte eines Vorgangs i generiert. Im Folgenden wird eine Anpassung der Algorithmen 4.3 und 4.4 beschrieben, wodurch die Listen $[\widetilde{d}_{ij}(W,t)]$ und $[\widehat{d}_{ij}(W,t)]$ für die Berechnung der indirekten zeitlichen Mindest- und Höchstabstände $\widetilde{d}_{ij}(W,\tau)$ und $\widehat{d}_{ij}(W,\tau)$ für alle Zeitpunkte $\tau \in W_i$ eines Vorgangs $i \in V$ verwendet werden können. Weiterhin wird durch die Anpassung der Algorithmen sichergestellt, dass kein W-zulässiger Startzeitpunkt eines Vorgangs eliminiert wird, sodass im Folgenden der W-Konsistenztest ohne eine vorangestellte Durchführung des T-Konsistenztests untersucht werden kann.

Die Aktualisierungsoperationen für die Startzeitbeschränkung W_i in den Zeilen 8 der Algorithmen 4.3 und 4.4 werden durch $d_{ijt} := ES'_j - t$ bzw. $d_{ijt} := LS'_j - t$ für alle Vorgänge $j \in V \setminus \{i\}$ ersetzt. Zusätzlich werden in beiden Algorithmen die Zeilen 22 angepasst. In Algorithmus 4.3 werden statt der Aktualisierung von W_i die Operationen $d_{ijt} := \infty$ für den Fall $t \leq \max W_i$ und $d_{ij,\max W_i} := \infty$ für den Fall $t < \max W_i$ über alle Vorgänge $j \in V \setminus \{i\}$ ausgeführt. Analog dazu ersetzen die Zuweisungen $d_{ijt} := -\infty$ für $t \geq \min W_i$ und $d_{ij,\min W_i} := -\infty$ für $t > \min W_i$ über alle Vorgänge $j \in V \setminus \{i\}$ die Aktualisierung von W_i in Algorithmus 4.4. Im weiteren Verlauf wird entsprechend $\widetilde{d}_{ij}(W,t) := \infty$ ($\widehat{d}_{ij}(W,t) := -\infty$) definiert, falls Schedule $ES(W,i,t)$ ($LS(W,i,t)$) nicht existiert.

Abbildung 4.4 zeigt für das Beispiel in Abbildung 4.2 unter der Annahme $ES_1 = 0$ den Verlauf der indirekten zeitlichen Mindest- und Höchstabstände für alle Startzeitpunkte $t \in W_1 = \{0, 1, \ldots, 9\}$ von Vorgang $i = 1$, wobei $\widetilde{d}_{13}(W,9) = \infty$ nicht abgebildet ist. Die Einträge der generierten Listen $[\widetilde{d}_{13}(W,t)]$ und $[\widehat{d}_{13}(W,t)]$ der angepassten Algorithmen 4.3 und 4.4 sind durch Quadrate gekennzeichnet. Wie aus der Abbildung 4.4 hervorgeht, können nun auch die indirekten zeitlichen Mindest- und Höchstabstände für die W-unzulässigen Startzeitpunkte 5, 6 und 9 von Vorgang $i = 1$ durch die Listen $[\widetilde{d}_{13}(W,t)]$ und $[\widehat{d}_{13}(W,t)]$ berechnet werden.

Eine Iteration des W-Konsistenztests ist in Algorithmus 4.12 dargestellt. Analog zum D-Konsistenztest wird zunächst eine Liste $[r_{ijkt}^{u,min}(W, \widetilde{D}, \widehat{D})]$ generiert, die sich aus den

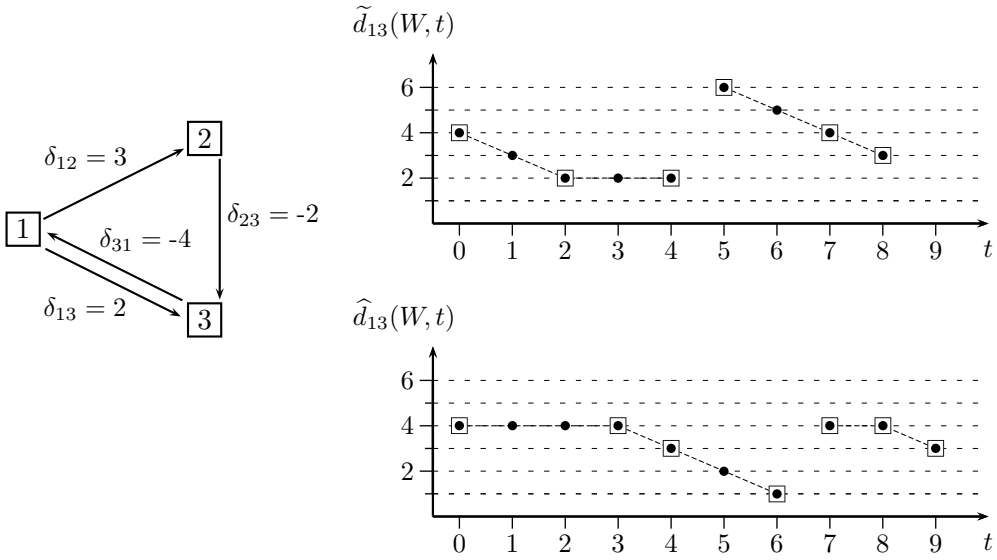

Abbildung 4.4: Verlauf der indirekten zeitlichen Mindest- und Höchstabstände

Mindestressourcenbelegungen

$$r_{ijkt}^{u,min}(W, \widetilde{D}, \widehat{D}) := \min\{r_{jk}^{u}(\tau) \mid \tau \in W_j \cap [t + \widetilde{d}_{ij}(W,t), t + \widehat{d}_{ij}(W,t)]\}$$

für ein Vorgangspaar $(i,j) \in V \times V_k$, $i \neq j$ und eine Ressource $k \in \mathcal{R}$ für eine Teilmenge aller Zeitpunkte $t \in \mathcal{H}$ zusammensetzt. Die Bestimmung der Liste $[r_{ijkt}^{u,min}(W, \widetilde{D}, \widehat{D})]$ wird wie in Algorithmus 4.6 vorgenommen, wobei zusätzlich die Verläufe der indirekten zeitlichen Mindest- und Höchstabstände, wie in Abbildung 4.4 angedeutet, berücksichtigt werden. Für die Erhöhung des betrachteten Zeitpunkts t in Algorithmus 4.6 wird außerdem eine maximale Zeitspanne bestimmt, die sicherstellt, dass die Verläufe der indirekten zeitlichen Mindest- und Höchstabstände innerhalb dieser Zeitspanne konstant sind. Weiterhin wird der Zeitpunkt t nur soweit erhöht, bis entweder $t = e_i(t)$ gilt oder ein Zeitpunkt erreicht wird, der am Ende einer Iteration von Algorithmus 4.3 oder zu Beginn einer Iteration von Algorithmus 4.4 in der Liste $[\widetilde{d}_{ij}(W,t)]$ bzw. $[\widehat{d}_{ij}(W,t)]$ abgespeichert wurde. Nach der Vergrößerung von Zeitpunkt t um die maximale Zeitspanne wird die Mindestressourcenbelegung r_{min}^{u} für den aktualisierten Zeitpunkt in der Liste $[r_{ijkt}^{u,min}(W, \widetilde{D}, \widehat{D})]$ abgespeichert. In der darauffolgenden Iteration werden dann die Werte der Variablen r_{min}^{u} und τ^{min} für den nächstgrößeren Zeitpunkt in W_i bestimmt und r_{min}^{u} für den aktuellen Zeitpunkt in der Liste $[r_{ijkt}^{u,min}(W, \widetilde{D}, \widehat{D})]$ abgelegt. Für die Bestimmung der Liste $[r_{ijkt}^{u,min}(W, \widetilde{D}, \widehat{D})]$ für die Vorgänge $i = 1$ und $j = 3$ in Abbildung 4.4 können die entsprechenden Zeitintervalle für die Erhöhung des Zeitpunkts t durch $([0,2],[2,3],[3,4])$, $([5,6])$, $([7,8])$, $([9,9])$ angegeben werden. Die eckigen Klammern stellen dabei die Zeitintervalle mit konstanten Verläufen von $\widetilde{d}_{13}(W,t)$ und $\widehat{d}_{13}(W,t)$ für die Vergrößerung des

Zeitpunkts t dar, wohingegen die runden Klammern alle Zeitintervalle einschließen, bis die Werte r^u_{min} und τ^{min}, wie beschrieben, neu bestimmt werden müssen. Für das Beispiel in Abbildung 4.4 werden entsprechend die Variablen r^u_{min} und τ^{min} für die Zeitpunkte 0, 5, 7 und 9 neu berechnet und Zeitpunkt t über die Zeitintervalle $[0,4]$, $[5,6]$, $[7,8]$ und $[9,9]$ unter Berücksichtigung der Verläufe der Mindest- und Höchstabstände $\widetilde{d}_{13}(W,t)$ und $\widehat{d}_{13}(W,t)$ vergrößert. Da bei einer Erhöhung von t auf einem Zeitintervall mit fallendem Verlauf des indirekten zeitlichen Höchstabstands $t + \widehat{d}_{ij}(W,t)$ konstant bleibt, kann r^u_{min} für alle Zeitpunkte des betrachteten Zeitintervalls nicht kleiner werden, sodass τ' (t') nicht berechnet wird. Das Gleiche gilt für die Berechnung von τ'' (t'') bei einem fallenden Verlauf des indirekten zeitlichen Mindestabstands, da $t + \widetilde{d}_{ij}(W,t)$ konstant bleibt, wodurch r^u_{min} durch die Erhöhung von t über alle Zeitpunkte des betrachteten Zeitintervalls nicht größer werden kann. Entsprechend kann der Zeitpunkt t über jedes Zeitintervall mit fallendem Verlauf des Mindest- und des Höchstabstands ohne Anpassung der Variablen r^u_{min} und τ^{min} vergrößert werden. Abschließend ist für die Anpassung von Algorithmus 4.6 für die Bestimmung der Liste $[r^{u,min}_{ijkt}(W,\widetilde{D},\widehat{D})]$ zu beachten, dass für jedes Zeitintervall mit konstanten Verläufen der Mindest- und Höchstabstände der kleinste und größte Zeitpunkt τ, für den $\widetilde{d}_{ij}(W,\tau) > \widehat{d}_{ij}(W,\tau)$ gilt, mit $r^u_{min} = \infty$ in der Liste $[r^{u,min}_{ijkt}(W,\widetilde{D},\widehat{D})]$ abgespeichert wird. Dadurch wird erreicht, dass der W-Konsistenztest jeden Zeitpunkt $t \in W_i$ eines Vorgangs $i \in V$ eliminiert, der die Bedingung des W-Konsistenztests erfüllt, wobei $\min \emptyset := \infty$ vorausgesetzt wird. Diese Eigenschaft des W-Konsistenztests kann daraus abgeleitet werden, dass für jeden Zeitpunkt $t \in W_i$ mit $\widetilde{d}_{ij}(W,t) \leq \widehat{d}_{ij}(W,t)$ die Bedingung $[t + \widetilde{d}_{ij}(W,t), t + \widehat{d}_{ij}(W,t)] \cap W_j \neq \emptyset$ erfüllt ist, da $t + \widetilde{d}_{ij}(W,t) \in W_j$ und $t + \widehat{d}_{ij}(W,t) \in W_j$ für $\widetilde{d}_{ij}(W,t) \neq \infty$ und $\widehat{d}_{ij}(W,t) \neq -\infty$ aus der Definition der Mindest- und Höchstabstände folgt.

Algorithmus 4.12: W-Konsistenztest

Input: Startzeitbeschränkung W
Output: $\gamma^{\mathrm{W}}(W)$

```
1 forall i ∈ V do
2 |  forall k ∈ R do
3 |  |  forall j ∈ V_k \ {i} do
4 |  |  |  Bestimme Liste [r^{u,min}_{ijkt}(W, D̃, D̂)]
5 |  |  Erzeuge Liste [r^{c,min}_{ikt}(W, D̃, D̂)]
6 Aktualisiere W (gemäß ∃k ∈ R : r^{c,min}_{ikt}(W, D̃, D̂) > R_k  ⇒  W_i := W_i \ {t})
7 return W
```

Nach der Berechnung der Listen $[r^{u,min}_{ijkt}(W,\widetilde{D},\widehat{D})]$ für alle Vorgänge $j \in V_k \setminus \{i\}$ wird analog zum D-Konsistenztest eine Liste $[r^{c,min}_{ikt}(W,\widetilde{D},\widehat{D})]$ generiert, die für eine Teilmenge

aller Zeitpunkte $t \in \mathcal{H}$ die Mindestinanspruchnahmen

$$r_{ikt}^{c,min}(W, \widetilde{D}, \widehat{D}) := r_{ik}^{c}(t) + \sum_{j \in V_k \setminus \{i\}} r_{ijkt}^{c,min}(W, \widetilde{D}, \widehat{D})$$

einer Ressource $k \in \mathcal{R}$ in Abhängigkeit vom Startzeitpunkt t des Vorgangs $i \in V$ beinhaltet. Mit Hilfe der Listen $[r_{ikt}^{c,min}(W, \widetilde{D}, \widehat{D})]$ über alle Ressourcen $k \in \mathcal{R}$ wird schließlich die Startzeitbeschränkung W_i für jeden Vorgang $i \in V$ gemäß der Bedingung des W-Konsistenztests aktualisiert, sodass nach Ablauf des Algorithmus 4.12 die aktualisierte Startzeitbeschränkung $\gamma^{\mathrm{W}}(W)$ ausgegeben wird.

Satz 4.9. *Algorithmus 4.12 hat eine Zeitkomplexität von* $\mathcal{O}(|V|^3(\mathcal{B}^2|\mathcal{R}| + \mathcal{I}^2))$.

Beweis. Zunächst wird die Zeitkomplexität für den angepassten Algorithmus 4.6 zur Bestimmung der Liste $[r_{ijkt}^{u,min}(W, \widetilde{D}, \widehat{D})]$ betrachtet. Nach jeder Iteration des angepassten Algorithmus wird entweder eine der Bedingungen, die im Beweis von Satz 4.7 für den Wert der Variablen τ^{min} genannt wurden, erfüllt oder Zeitpunkt t mindestens so weit vergrößert, dass entweder $t = e_i(t)$ gilt oder ein nächstgrößerer Zeitpunkt in einer der Listen $[\widetilde{d}_{ij}(W, t)]$ oder $[\widehat{d}_{ij}(W, t)]$ erreicht wird. Entsprechend folgt eine maximale Anzahl an $\mathcal{O}(\mathcal{B} + \mathcal{I}_k)$ Iterationen für den angepassten Algorithmus 4.6, da in den Listen $[\widetilde{d}_{ij}(W, t)]$ und $[\widehat{d}_{ij}(W, t)]$ höchstens $\mathcal{O}(\mathcal{B}+1)$ Zeitpunkte abgespeichert sind (vgl. Beweis von Satz 4.4) und Zeitpunkt t streng monoton über alle Iterationen steigt. Da weiterhin $\mathcal{O}(\mathcal{B}_j + \mathcal{I}_k)$ Operationen in jeder Iteration ausgeführt werden, ergibt sich schließlich eine Zeitkomplexität von $\mathcal{O}(\mathcal{B}^2+\mathcal{I}_k^2)$ für den angepassten Algorithmus 4.6. Analog zum Beweis von Satz 4.8 wird die Liste $[r_{ikt}^{c,min}(W, \widetilde{D}, \widehat{D})]$ für einen Vorgang $i \in V$ und eine Ressource $k \in \mathcal{R}$ mit einer Zeitkomplexität von $\mathcal{O}(|V|^2(\mathcal{B} + \mathcal{I}_k))$ generiert. Entsprechend sind $\mathcal{O}(|V|^2(\mathcal{B}^2|\mathcal{R}| + \mathcal{I}^2))$ Operationen für die Bestimmung aller Listen $[r_{ijkt}^{u,min}(W, \widetilde{D}, \widehat{D})]$ und $[r_{ikt}^{c,min}(W, \widetilde{D}, \widehat{D})]$ für einen Vorgang $i \in V$ erforderlich, sodass Algorithmus 4.12 mit einer Zeitkomplexität von $\mathcal{O}(|V|^3(\mathcal{B}^2|\mathcal{R}|+\mathcal{I}^2))$ ausgeführt wird (vgl. Beweis von Satz 4.8). □

Abschließend werden die Zusammenhänge zwischen den in diesem Abschnitt beschriebenen Konsistenztests untersucht. Zur Vereinfachung wird im Folgenden $W \subseteq W'$ für zwei Startzeitbeschränkungen W und W' geschrieben, falls $W_i \subseteq W'_i$ für alle Vorgänge $i \in V$ gilt. Zusätzlich geben $\gamma^{\mathrm{T}}(W)$, $\gamma^{\mathrm{TB}}(W)$ und $\gamma^{\mathrm{RB}}(W)$ die Startzeitbeschränkungen nach der Ausführung einer Iteration des T-, TB- und RB-Konsistenztests für eine beliebige Startzeitbeschränkung W an. Zunächst lassen sich die Zusammenhänge $\gamma^{\mathrm{T}}(W) \subseteq \gamma^{\mathrm{TB}}(W)$ und $\gamma^{\mathrm{W}}(W) \subseteq \gamma^{\mathrm{D}}(W) \subseteq \gamma^{\mathrm{RB}}(W)$ zwischen den Konsistenztests ableiten. Weiterhin kann festgestellt werden, dass jeder der betrachteten Konsistenztests in diesem Abschnitt in Anlehnung an Definition 2.7 in Dorndorf et al. (2000b) monoton ist, d. h., $W \subseteq W'$ impliziert $\gamma(W) \subseteq \gamma(W')$. Aus den Zusammenhängen zwischen

den Konsistenztests und aus der Monotonie ergeben sich schließlich die Zusammenhänge $W^{\mathrm{T}} \subseteq W^{\mathrm{TB}}$ und $W^{\mathrm{W}} \subseteq W^{\mathrm{D}} \subseteq W^{\mathrm{RB}}$ zwischen den Fixpunkten der einzelnen Konsistenztests für eine beliebige Startzeitbeschränkung W.

Bislang wurde für die Untersuchungen der Konsistenztests davon ausgegangen, dass jeder der Konsistenztests einzeln ausgeführt wird. Im Folgenden wird ein allgemeines Verfahren vorgestellt, das verschiedene Domain-Konsistenztests zur Reduktion der Startzeitpunkte einer gegebenen Startzeitbeschränkung W ausführt. Algorithmus 4.13 zeigt das entsprechende Verfahren, das in jeder Iteration alle Konsistenztests einer Menge Γ^{β} auf einer Startzeitbeschränkung W ausführt, bis entweder ein Fixpunkt erreicht oder die maximale Anzahl an Iterationen α überschritten wird. Das Verfahren entspricht Algorithmus 2.1 in Dorndorf et al. (2000b) mit einer zusätzlich vorgegebenen maximalen Anzahl an Iterationen, sodass Algorithmus 4.13 im Allgemeinen die Bestimmung eines Fixpunkts nicht garantiert. Die Ausgabe von Algorithmus 4.13 wird durch $\gamma_{\beta}^{\alpha}(W)$ beschrieben, wobei im weiteren Verlauf dieser Arbeit durch $\alpha = \infty$ angegeben wird, dass Algorithmus 4.13 einen Fixpunkt der Konsistenztests der Menge Γ^{β} bestimmt. Dabei ist zu beachten, dass aufgrund der Monotonie aller Konsistenztests aus Theorem 2.2 in Dorndorf et al. (2000b) folgt, dass der Fixpunkt einer Menge Γ^{β} immer eindeutig, d. h., unabhängig von der Reihenfolge der Ausführungen der Konsistenztests ist.

Algorithmus 4.13: Ablaufschema für Domain-Konsistenztests

Input: Startzeitbeschränkung W, Γ^{β}, α
Output: Aktualisierte Startzeitbeschränkung $\gamma_{\beta}^{\alpha}(W)$

```
1 repeat
2 |   W' := W
3 |   forall γ ∈ Γ^β do
4 |   |   W := γ(W)
5 until W = W' ∨ ∃i ∈ V : W_i = ∅
    oder die maximale Anzahl an Iterationen α erreicht ist
6 return W
```

Für die Branch-and-Bound-Verfahren in Kapitel 5 werden vier verschiedene Kombinationen aus Konsistenztests betrachtet, die durch die Bezeichner Γ^{B}, Γ^{T}, Γ^{D} und Γ^{W} angegeben werden. Die Mengen Γ^{B} und Γ^{T} setzen sich aus dem TB- und RB-Konsistenztest bzw. dem T- und RB-Konsistenztest zusammen. Die weiteren Mengen Γ^{D} und Γ^{W} beinhalten dagegen den TB- und D-Konsistenztest bzw. den T- und W-Konsistenztest. Im weiteren Verlauf dieser Arbeit werden zwei Varianten für die D- und W-Konsistenztests voneinander unterschieden, die entweder alle Ressourcen $k \in \mathcal{R}$ oder nur die Ressourcen

$k \in \mathcal{R}_i$ für einen Vorgang $i \in V$ in einer Iteration berücksichtigen. Zur Unterscheidung beider Varianten werden die Ausgaben von Algorithmus 4.13 für Γ^{D} und Γ^{W} jeweils durch $\gamma^{\alpha}_{\beta}[\mathcal{R}](W)$ oder $\gamma^{\alpha}_{\beta}[\mathcal{R}_i](W)$ gekennzeichnet.

4.3 Untere Schranken

Untere Schranken werden in Branch-and-Bound-Verfahren mit zu minimierender Zielfunktion dafür verwendet, um Teile des Suchbaums von weiteren Untersuchungen auszuschließen, wodurch die Berechnungszeit verkürzt bzw. eine vollständige Enumeration vermieden wird. Untere Schrankenwerte, die im Wurzelknoten des Suchbaums bestimmt werden, können als globale Schrankenwerte über alle Enumerationsknoten oder auch zur Abschätzung der Güte zulässiger Lösungen der Probleminstanz verwendet werden.

Im Folgenden werden zwei untere Schranken für die kürzeste Projektdauer über alle zulässigen Schedules innerhalb des Suchraums eines Enumerationsknotens $\mathcal{S}_T(W)$ eines der Branch-and-Bound-Verfahren, die in Kapitel 5 besprochen werden, vorgestellt. Die erste untere Schranke $LB0^{\pi}$ entspricht dem Zielfunktionswert einer optimalen Lösung des Problems (PS(W)), sodass $LB0^{\pi} = ES_{n+1}(W)$ gilt. Wie in Abschnitt 4.1.1 gezeigt wurde, kann $LB0^{\pi}$ durch Algorithmus 4.1 mit einer Zeitkomplexität von $\mathcal{O}(|V||E|(\mathcal{B}+1))$ berechnet werden. Dabei ist zu beachten, dass die Bestimmung von $ES(W)$ Teil der Enumeration aller Branch-and-Bound-Verfahren ist, sodass $LB0^{\pi}$ keinen zusätzlichen Berechnungsaufwand erfordert. Die zweite untere Schranke LBD^{π} entspricht einer sogenannten destruktiven (zurückweisenden) unteren Schranke (vgl. Klein und Scholl, 1999), für die eine hypothetische maximale Projektdauer d ausgehend von einem unteren Grenzwert (z. B. $LB0^{\pi}$) solange erhöht wird, bis schließlich nicht mehr gezeigt werden kann, dass d die Existenz eines zulässigen Schedules im Suchraum ausschließt. Algorithmus 4.14 zeigt das Verfahren für die Bestimmung der destruktiven unteren Schranke LBD^{π}. Das Verfahren ist dabei an den Algorithmus 1 in Franck et al. (2001b) angelehnt.

Algorithmus 4.14 bestimmt für jeden Enumerationsknoten die kleinste hypothetische maximale Projektdauer auf einem vorgegebenen Intervall $[LB^{start}, UB^{start}]$, für die die Existenz eines zulässigen Schedules nicht ausgeschlossen werden kann. Im Folgenden wird vorausgesetzt, dass $LB^{start} \geq 0$ einer unteren und $UB^{start} \leq \bar{d}$ einer oberen Schranke für die kürzeste Projektdauer aller zulässigen Schedules im Suchraum eines Enumerationsknotens entspricht. Weiterhin wird angenommen, dass die Bedingungen $\widetilde{\mathcal{S}}_T(W, n+1, LB^{start}) \neq \emptyset$ und $ES_{n+1}(W, n+1, LB^{start}) \leq UB^{start}$ erfüllt sind, da ansonsten bereits bekannt wäre, dass keine zulässige Lösung $S \in \mathcal{S}(W)$ existiert. Algorithmus 4.14 führt

Algorithmus 4.14: Destruktive untere Schranke LBD^{π}

Input: Startzeitbeschränkung W, LB^{start}, UB^{start}
Output: LBD^{π}

```
1  LB^d := LB^start, UB^d := UB^start
2  ES' := ES(W, n+1, LB^start)
3  while LB^d ≤ UB^d do
4  |  d := ⌈(LB^d + UB^d)/2⌉
5  |  LS' := LS(W, n+1, d)
6  |  if ∃k ∈ R : r_k^{c,min}(W, ES', LS') > R_k then
7  |  |  if d = UB^start then
8  |  |  |  terminate
9  |  |  LB^d := d + 1
10 |  else
11 |  |  UB^d := d − 1
12 LBD^π := LB^d
13 return LBD^π
```

eine binäre Suche über alle ganzzahligen Zeitpunkte im Intervall $[LB^{start}, UB^{start}]$ durch. In jeder Iteration wird ein Intervall $[LB^d, UB^d]$ betrachtet, beginnend mit $LB^d := LB^{start}$ und $UB^d := UB^{start}$ in der ersten Iteration des Verfahrens. Für jedes Intervall $[LB^d, UB^d]$ wird eine hypothetische maximale Projektdauer $d := \lceil (LB^d + UB^d)/2 \rceil$ bestimmt. Falls gezeigt werden kann, dass für die maximale Projektdauer d kein zulässiger Schedule existiert, kann abgeleitet werden, dass die kürzeste Projektdauer über alle zulässigen Schedules $S \in \mathcal{S}(W)$ größer als d ist. Im Fall $d = UB^{start}$ terminiert das Verfahren, da keine zulässige Lösung im Suchraum des betrachteten Enumerationsknotens existiert. Andernfalls wird $LB^d := d + 1$ gesetzt, sodass das Intervall $[d + 1, UB^d]$ in der nächsten Iteration betrachtet wird. Für den Fall, dass d nicht zurückgewiesen wird, da eine zulässige Lösung für d existieren könnte, wird das Intervall $[LB^d, d - 1]$ in der nächsten Iteration untersucht, indem $UB^d := d - 1$ gesetzt wird. Das beschriebene Vorgehen wird so lange ausgeführt, bis $LB^d > UB^d$ gilt. Am Ende des Verfahrens wird LB^d schließlich als destruktive untere Schranke LBD^{π} ausgegeben.

Als Nächstes wird beschrieben, wie in Algorithmus 4.14 überprüft wird, ob eine maximale Projektdauer d die Existenz eines zulässigen Schedules im Suchraum eines Enumerationsknotens ausschließt. Gegeben sei dafür eine hypothetische maximale Projektdauer $d \in [LB^{start}, UB^{start}] \cap \mathbb{Z}$ in einer beliebigen Iteration von Algorithmus 4.14. Dann wird zunächst die Mindestinanspruchnahme für jede Ressource $k \in \mathcal{R}$ und jeden Vorgang $i \in V_k$ über alle Startzeitpunkte $t \in W_i \cap [ES'_i, LS'_i]$ berechnet. ES'_i und LS'_i entsprechen

dabei dem frühesten und spätesten W-zulässigen Startzeitpunkt von Vorgang i, unter der Annahme, dass die Projektdauer nicht kleiner als LB^{start} und nicht größer als d ist. Aus den entsprechenden Mindestressourceninanspruchnahmen

$$r_{ik}^{c,min}(W, ES'_i, LS'_i) := \min\{r_{ik}^{c}(\tau) \mid \tau \in W_i \cap [ES'_i, LS'_i]\}$$

mit $ES' := ES(W, n+1, LB^{start})$ und $LS' := LS(W, n+1, d)$ wird schließlich die Gesamtmindestinanspruchnahme

$$r_{k}^{c,min}(W, ES', LS') := \sum_{i \in V_k} r_{ik}^{c,min}(W, ES'_i, LS'_i)$$

für jede Ressource $k \in \mathcal{R}$ berechnet. Für den Fall, dass die Gesamtmindestinanspruchnahme $r_{k}^{c,min}(W, ES', LS')$ mindestens einer Ressource $k \in \mathcal{R}$ die Kapazität R_k übersteigt, kann für die maximale Projektdauer d kein zulässiger Schedule $S \in \mathcal{S}(W)$ existieren, sodass d zurückgewiesen wird. Andernfalls kann die Existenz einer zulässigen Lösung im Suchraum des Enumerationsknotens für die maximale Projektdauer d nicht ausgeschlossen werden, sodass d nicht zurückgewiesen wird.

Satz 4.10. *Algorithmus 4.14 hat eine Zeitkomplexität von* $\mathcal{O}(\log(\bar{d}+1)(|V||E|(\mathcal{B}+1) + |V|\mathcal{I} + |\mathcal{R}|\mathcal{B}))$.

Beweis. Zunächst folgt aus der binären Suche auf dem Zeitintervall $[LB^{start}, UB^{start}]$ eine maximale Anzahl an $\mathcal{O}(\log_2(\bar{d}+1))$ Iterationen für Algorithmus 4.14. In jeder der Iterationen wird LS' mit einer Zeitkomplexität von $\mathcal{O}(|V||E|(\mathcal{B}+1))$ durch Algorithmus 4.2 aktualisiert. Da analog zum RB-Konsistenztest die Mindestressourceninanspruchnahmen über alle Ressourcen und Vorgänge $r_{ik}^{c,min}(W, ES'_i, LS'_i)$ in jeder Iteration mit einer Zeitkomplexität von $\mathcal{O}(|V|\mathcal{I} + |\mathcal{R}|\mathcal{B})$ berechnet werden können, folgt schließlich eine Zeitkomplexität von $\mathcal{O}(\log(\bar{d}+1)(|V||E|(\mathcal{B}+1) + |V|\mathcal{I} + |\mathcal{R}|\mathcal{B}))$ für Algorithmus 4.14. □

Kapitel 5

Branch-and-Bound-Verfahren

Die Branch-and-Bound-Methode stellt eine allgemeine Lösungsmethodik bzw. ein Metaverfahren zur exakten Lösung von Optimierungsproblemen dar. Die grundlegende Verfahrensidee besteht dabei darin, den zulässigen Bereich einer Probleminstanz schrittweise zu zerlegen, um unter Zuhilfenahme von Schrankenwerten Teilmengen des zulässigen Bereichs auszuschließen, die entweder keine Lösung oder keine bessere als eine beste bekannte Lösung enthalten. Ein Branch-and-Bound-Verfahren setzt sich im Allgemeinen aus verschiedenen Komponenten zusammen, wobei die Basis ein Enumerationsschema bildet. Im weiteren Verlauf dieser Arbeit soll unter einem Enumerationsschema ein Verfahren verstanden werden, das für eine Instanz einer gegebenen Problemstellung entweder die Unlösbarkeit beweist oder andernfalls eine Menge zulässiger Lösungen generiert, von denen mindestens eine optimal ist. Ein Enumerationsschema kann i. d. R. als Wurzelbaum (vgl. Neumann und Morlock, 2002, Abschnitt 2.1.3) bzw. Enumerationsbaum dargestellt werden, wobei der Wurzelknoten den Beginn und jeder weitere Knoten auf einem Weg im Suchbaum das Hinzufügen einer zusätzlichen Restriktion repräsentiert. Der Enumerationsverlauf bzw. die schrittweise Generierung des Enumerationsbaums wird durch die Suchstrategie des Branch-and-Bound-Verfahrens festgelegt. Um eine vollständige Enumeration zu vermeiden bzw. Teile des Enumerationsbaums vorzeitig auszuschließen, werden Schrankenwerte zur Abschätzung der Güte bestmöglicher Lösungen im zulässigen Bereich des jeweiligen Enumerationsknotens bestimmt. Zur weiteren Einschränkung der Enumeration können zudem auch Konsistenztests oder Techniken zur Redundanzvermeidung eingesetzt werden.

In den folgenden Abschnitten werden Branch-and-Bound-Verfahren für das RCPSP/ max-π entwickelt, die zwei unterschiedlichen Kategorien zugeordnet werden können. Im Bereich der ressourcenbeschränkten Projektplanung werden im Allgemeinen relaxations- und konstruktionsbasierte Enumerationsschemata voneinander unterschieden.

Dabei wird in einem relaxationsbasierten Enumerationsansatz einer Relaxation der betrachteten Problemstellung in jedem Verzweigungsschritt eine zusätzliche Restriktion hinzugefügt, wohingegen ein konstruktionsbasierter Ansatz die Vorgänge eines Projekts sukzessiv über den Enumerationsverlauf einplant. Die Branch-and-Bound-Verfahren, die in den Abschnitten 5.1 und 5.2 eingeführt werden, können der ersten Kategorie, den relaxationsbasierten Ansätzen, zugeordnet werden, wohingegen in Abschnitt 5.3 ein konstruktionsbasierter Ansatz betrachtet wird.

5.1 Ein relaxationsbasierter Ansatz

In Abschnitt 5.1.1 wird zunächst das Enumerationsschema für das relaxationsbasierte Branch-and-Bound-Verfahren vorgestellt, das in jedem Enumerationsschritt der Ressourcenrelaxation von Problem (P) eine Einschränkung für die Ressourcenbelegung eines Vorgangs hinzufügt. Zur Vermeidung von Redundanzen im Enumerationsverlauf werden im Abschnitt 5.1.2 Dominanzregeln entwickelt, durch die Teile des Enumerationsbaums von weiteren Untersuchungen ausgeschlossen werden können. Abschnitt 5.1.3 zeigt weiterhin, wie das Enumerationsschema in Abschnitt 5.1.1 angepasst werden kann, sodass Teilmengen des zulässigen Bereichs der Ressourcenrelaxation nicht mehrfach im Enumerationsverlauf untersucht werden. In Abschnitt 5.1.4 wird schließlich das relaxationsbasierte Branch-and-Bound-Verfahren beschrieben, das aus dem Enumerationsschema in Abschnitt 5.1.1 entwickelt wird.

Es ist zu beachten, dass die Inhalte dieses Abschnitts bereits in der Arbeit Watermeyer und Zimmermann (2020), die im Rahmen der Promotion des Autors der vorliegenden Dissertation entstanden ist, veröffentlicht wurden.

5.1.1 Enumerationsschema

Im Folgenden wird ein relaxationsbasiertes Enumerationsschema für das RCPSP/max-π entwickelt, das ausgehend von der Ressourcenrelaxation von Problem (P) die Ressourcenbelegungen der Vorgänge des Projekts im Enumerationsverlauf solange einschränkt, bis entweder die optimale Lösung eines Enumerationsknotens zulässig ist oder keine Lösung mehr im Suchraum existiert. Jedem Knoten im Enumerationsbaum wird dazu eine Startzeitbeschränkung W zugeordnet, die sich, wie nachfolgend gezeigt wird, aus den Einschränkungen der Ressourcenbelegungen ergibt, wobei die Startzeitbeschränkung des

Wurzelknotens alle zeitzulässigen Startzeitpunkte der Vorgänge umfasst. In jedem Knoten des Enumerationsbaums wird das Problem (PS(W)) gelöst bzw. mit $S = \min \mathcal{S}_T(W)$ ein optimaler Schedule im Suchraum $\mathcal{S}_T(W)$ bestimmt, falls $\mathcal{S}_T(W) \neq \emptyset$ gilt. Für den Fall, dass Schedule S nicht zulässig ist ($S \notin \mathcal{S}$), liegt mindestens eine Ressource $k \in \mathcal{R}$ vor, deren Kapazität R_k durch die Inanspruchnahme $r_k^c(S)$ überschritten wird. Um diesen Ressourcenkonflikt im weiteren Enumerationsverlauf aufzulösen, wird die aktuelle Belegung eines Vorgangs $i \in V_k$ mit $r_{ik}^u(S_i) > 0$ um eine Einheit reduziert. Dieses Vorgehen wird solange fortgeführt, bis schließlich für jedes Blatt im Enumerationsbaum entweder ein zulässiger Schedule $S = \min \mathcal{S}_T(W)$ bestimmt oder $\mathcal{S}(W) = \emptyset$ gezeigt werden konnte.

Das entsprechende Enumerationsschema bzw. die Generierung des Enumerationsbaums wird in Algorithmus 5.1 beschrieben. Zu Beginn des Verfahrens werden zunächst die frühesten und spätesten zeitzulässigen Startzeitpunkte ES_i und LS_i aller Vorgänge $i \in V$ durch ein Label-Correcting-Verfahren (s. z. B. Ahuja et al., 1993, Abschnitt 5.4) bestimmt, falls mindestens ein zeitzulässiger Schedule existiert, d. h., $\mathcal{S}_T \neq \emptyset$ gilt. Im Fall $\mathcal{S}_T = \emptyset$ terminiert das Verfahren, da für die betrachtete Instanz kein zulässiger Schedule existiert. Im Folgenden sei daher $\mathcal{S}_T \neq \emptyset$ angenommen. Zunächst wird die Startzeitbeschränkung $W = (W_i)_{i \in V}$ mit $W_i := [ES_i, LS_i] \cap \mathbb{Z}$ für alle Vorgänge $i \in V$ der Menge Ω, die alle im Enumerationsverlauf noch nicht betrachteten Knoten enthält, hinzugefügt. Im Anschluss daran wird die Menge Φ, die alle im Enumerationsverlauf generierten Schedules abspeichert, durch $\Phi := \emptyset$ initialisiert.

In jeder Iteration von Algorithmus 5.1 wird eine Startzeitbeschränkung W aus der Menge Ω entnommen und das Problem (PS(W)) gelöst bzw. der Schedule $S = \min \mathcal{S}_T(W)$ bestimmt. Für den Fall, dass Schedule S zulässig ist, d. h., $r_k^c(S) \leq R_k$ für alle $k \in \mathcal{R}$ gilt, wird Schedule S der Menge Φ hinzugefügt. Andernfalls existiert mindestens eine Konfliktressource $k \in \mathcal{R}^c(S) := \{k \in \mathcal{R} \mid r_k^c(S) > R_k\}$, deren Kapazität durch die Inanspruchnahme $r_k^c(S)$ überschritten wird. In diesem Fall wird der Suchraum $\mathcal{S}_T(W)$ des Enumerationsknotens zerlegt, indem die Ressourcenbelegungen aller Vorgänge $i \in V_k$, die Ressource k in Anspruch nehmen, eingeschränkt werden. Im Folgenden gibt $\bar{u}_{ik}$ eine obere Schranke für die Belegung einer Ressource $k \in \mathcal{R}$ durch einen Vorgang $i \in V$ an, die einem Knoten im Enumerationsverlauf hinzugefügt wird, sodass die Bedingung $r_{ik}^u(S_i) \leq \bar{u}_{ik}$ durch jeden Schedule im Suchraum des Enumerationsknotens eingehalten wird. $\bar{u}_{ik}$ wird nachfolgend auch als Belegungsbeschränkung einer Ressource $k \in \mathcal{R}$ für einen Vorgang $i \in V$ bezeichnet. Weiterhin gibt $W_{ik}(\bar{u}_{ik}) := \{\tau \in [ES_i, LS_i] \cap \mathbb{Z} \mid r_{ik}^u(\tau) \leq \bar{u}_{ik}\}$ die durch $\bar{u}_{ik}$ induzierte Startzeitbeschränkung von Vorgang $i \in V$ und Ressource $k \in \mathcal{R}$ an, die alle zeitzulässigen Startzeitpunkte von Vorgang i mit einer maximalen Belegung der

Algorithmus 5.1: Relaxationsbasiertes Enumerationsschema

Input: RCPSP/max-π-Instanz
Output: Menge Φ aller generierten Schedules

```
1  Bestimme die Schedules ES und LS
2  if S_T = ∅ then terminate (S = ∅)
3  W_i := [ES_i, LS_i] ∩ ℤ für alle i ∈ V
4  Ω := {W},        Φ := ∅
5  while Ω ≠ ∅ do
6  |  Entferne W aus der Menge Ω
7  |  S := min S_T(W)
8  |  if S ∈ S_R then
9  |  |  Φ := Φ ∪ {S}
10 |  else
11 |  |  Wähle eine Ressource k ∈ R mit r^c_k(S) = Σ_{i∈V} r^u_ik(S_i) r^d_ik > R_k
12 |  |  forall i ∈ {j ∈ V_k | r^u_jk(S_j) > 0} do
13 |  |  |  W' := W,        W'_i := W'_i ∩ W_ik(r^u_ik(S_i) − 1)
14 |  |  |  if S_T(W') ≠ ∅ then
15 |  |  |  |  Ω := Ω ∪ {W'}
16 return Φ
```

Ressource k in Höhe von $\bar{u}_{ik}$ enthält. Die Zerlegung des Suchraums $\mathcal{S}_T(W)$ wird basierend auf Satz 5.1 wie folgt ausgeführt. Für jeden Vorgang $i \in V_k$ mit $r^u_{ik}(S_i) > 0$ wird die obere Schranke für die maximale Ressourcenbelegung $\bar{u}_{ik}$ auf $r^u_{ik}(S_i) - 1$ gesetzt, sodass alle Zeitpunkte $t \in W_i$ mit $r^u_{ik}(t) \geq r^u_{ik}(S_i)$ aus W_i entfernt werden. Die Startzeitbeschränkung W' des direkten Nachfolgerknotens, der Vorgang $i \in V_k$ zugeordnet ist, wird durch $W' := W$ und $W'_i := W_i \cap W_{ik}(r^u_{ik}(S_i)-1)$ bestimmt. Falls $\mathcal{S}_T(W') \neq \emptyset$ gilt, wird W' der Menge Ω hinzugefügt und in den nachfolgenden Iterationen weiter untersucht. Nachdem schließlich alle Enumerationsknoten betrachtet wurden, d. h., sobald $\Omega = \emptyset$ gilt, gibt Algorithmus 5.1 die Menge Φ zurück, die alle im Enumerationsverlauf generierten Schedules enthält.

Satz 5.1. *Gegeben sei $\sum_{i \in V} u_{ik}\, r^d_{ik} > R_k$ mit $u_{ik} \in \{0, 1, \ldots, p_i\}$ für eine Ressource $k \in \mathcal{R}$. Dann erfüllt jeder zulässige Schedule $S \in \mathcal{S}$ die Bedingung $S_i \in W_{ik}(u_{ik} - 1)$ für mindestens einen Vorgang $i \in V_k$.*

Beweis. Sei S ein beliebiger zulässiger Schedule, der die Bedingung $S_i \in W_{ik}(u_{ik} - 1)$ für keinen Vorgang $i \in V_k$ erfüllt. Dann folgt $\sum_{i \in V_k} r^u_{ik}(S_i)\, r^d_{ik} \geq \sum_{i \in V_k} u_{ik}\, r^d_{ik} > R_k$ aus $r^u_{ik}(S_i) \geq u_{ik}$ für jeden Vorgang $i \in V_k$, sodass sich ein Widerspruch zur Annahme $S \in \mathcal{S}$ ergibt. □

Die Korrektheit von Algorithmus 5.1 ergibt sich aus Satz 5.2, da im Enumerationsverlauf ausschließlich zulässige Schedules generiert werden ($\Phi \subseteq \mathcal{S}$) und $\mathcal{OS} \neq \emptyset \Leftrightarrow \mathcal{S} \neq \emptyset$ aus

der Endlichkeit des zulässigen Bereichs $\mathcal{S}$ folgt. Weiterhin gibt Lemma 5.1 eine obere Schranke für die Anzahl der generierten Enumerationsknoten in Algorithmus 5.1 an, woraus die Endlichkeit des Enumerationsverfahrens folgt.

Lemma 5.1. *Algorithmus 5.1 generiert maximal* $|V|^{|V||\mathcal{R}|\overline{p}}$ *Enumerationsknoten.*

Beweis. Zunächst kann die Belegung einer Ressource durch einen Vorgang höchstens $\overline{p} := \max_{i \in V} p_i$ mal im Enumerationsverlauf eingeschränkt werden, sodass der Suchbaum eine maximale Tiefe von $|V||\mathcal{R}|\overline{p}$ hat. Da weiterhin höchstens $|V|$ Nachfolgerknoten in jedem Verzweigungsschritt generiert werden, folgt schließlich eine maximale Anzahl an $|V|^{|V||\mathcal{R}|\overline{p}}$ Knoten, die im Enumerationsverlauf erzeugt werden. □

Satz 5.2. *Algorithmus 5.1 ist vollständig, d. h.,* $\mathcal{OS} \neq \emptyset \Rightarrow \Phi \cap \mathcal{OS} \neq \emptyset$.

Beweis. Zunächst gilt $\mathcal{S}_T(W) = \mathcal{S}_T \supseteq \mathcal{S}$ für den Wurzelknoten des Enumerationsbaums. Weiterhin folgt aus Satz 5.1, dass durch die Zerlegung des Suchraums $\mathcal{S}_T(W)$ eines Enumerationsknotens kein zulässiger Schedule $S \in \mathcal{S}(W)$ ausgeschlossen wird. Entsprechend kann jeder zulässige Schedule $S \in \mathcal{S}$ nach Abschluss der Enumeration mindestens einem Blatt W im Enumerationsbaum zugeordnet werden ($S \in \mathcal{S}_T(W)$). Aus der Endlichkeit des Verfahrens nach Lemma 5.1 und daraus, dass Schedule $S = \min \mathcal{S}_T(W)$ die Projektdauer auf $\mathcal{S}_T(W)$ minimiert, folgt schließlich aus $\mathcal{OS} \neq \emptyset \Rightarrow \mathcal{S} \neq \emptyset$ die Vollständigkeit des Verfahrens. □

5.1.2 Dominanzregeln

Im Allgemeinen werden Dominanzregeln in Branch-and-Bound-Verfahren dazu verwendet, um Redundanzen im Enumerationsverlauf zu detektieren, wodurch Teile des Suchbaums von weiteren Untersuchungen ausgeschlossen werden können. Im weiteren Verlauf dieser Arbeit wird von einer Redundanz im Enumerationsverlauf gesprochen, wenn eine beliebige Teilmenge des gesamten Suchraums ($\mathcal{S}_T$) in unterschiedlichen Bereichen des Enumerationsbaums untersucht wird. Für eine Dominanzregel wird i. d. R. eine Bedingung zwischen zwei voneinander nicht erreichbaren Knoten im Enumerationsbaum überprüft. Falls die Bedingung erfüllt wird, kann daraus abgeleitet werden, dass für jede zulässige Lösung im Suchraum eines der Knoten (redundanter Knoten) mindestens eine zulässige Lösung mit einem gleichwertigen oder besseren Zielfunktionswert im Suchraum des anderen Knotens (dominanter Knoten) existiert. Entsprechend können der redundante Knoten und alle seine Nachfolger aus dem Enumerationsbaum entfernt (ausgelotet)

werden, falls die Bedingung der betrachteten Dominanzregel erfüllt ist und die vollständige Untersuchung des Suchraums des dominanten Knotens sichergestellt ist. Um die Dominanzregeln für die Anwendung im Branch-and-Bound-Algorithmus in Abschnitt 5.1.4 zu untersuchen, wird im weiteren Verlauf davon ausgegangen, dass die Konsistenztests aus Abschnitt 4.2 zur Einschränkung des Suchraums für jeden Enumerationsknoten verwendet werden.

Im Folgenden werden zwei unterschiedliche Dominanzregeln für das relaxationsbasierte Enumerationsschema aus Abschnitt 5.1.1 vorgestellt. Beide Dominanzregeln haben gemeinsam, dass sie für zwei voneinander nicht erreichbare Knoten W' und W'' im Enumerationsbaum überprüfen, ob $\mathcal{S}(W') \subseteq \mathcal{S}(W'')$ gilt. Entsprechend impliziert die Erfüllung der zugehörigen Bedingung für jede der Dominanzregeln, dass Knoten W' durch Knoten W'' dominiert wird, sodass der redundante Knoten W' und alle Nachfolgerknoten von den weiteren Untersuchungen ausgeschlossen werden können.

Für die erste Dominanzregel wird jedem Enumerationsknoten eine sogenannte Ressourcenbeschränkungsmatrix $\bar{U} := (\bar{u}_{ik})_{i \in V, k \in \mathcal{R}}$ zugeordnet, um die Ressourcenbeschränkungen $\bar{u}_{ik}$, die im Enumerationsverlauf jedem Knoten zur Einschränkung des Suchraums hinzugefügt werden, abzuspeichern. Zu Beginn des Verfahrens wird $\bar{U}$ zunächst im Wurzelknoten durch $\bar{u}_{ik} := p_i$ für alle $i \in V$ und $k \in \mathcal{R}$ initialisiert und im weiteren Enumerationsverlauf durch die hinzugefügten Ressourcenbeschränkungen $\bar{u}_{ik}$ in jedem Knoten aktualisiert. Um für jeden Enumerationsknoten alle zeitzulässigen Startzeitpunkte eines Vorgangs zu beschreiben, die alle hinzugefügten Ressourcenbeschränkungen erfüllen, werden zusätzliche Bezeichner eingeführt. Der Vektor $W(\bar{U}) := (W_i(\bar{U}))_{i \in V}$ ist die sogenannte $\bar{U}$-induzierte Startzeitbeschränkung mit $W_i(\bar{U}) := \bigcap_{k \in \mathcal{R}} W_{ik}(\bar{u}_{ik})$ als der $\bar{U}$-induzierten Startzeitbeschränkung von Vorgang $i \in V$. Zur Vereinfachung wird nachfolgend $W' \subseteq W''$ anstatt $W'_i \subseteq W''_i$ für alle $i \in V$ sowie $\bar{U}' \leq \bar{U}''$ an Stelle von $\bar{u}'_{ik} \leq \bar{u}''_{ik}$ für alle $i \in V$ und $k \in \mathcal{R}$ geschrieben.

Die erste Regel, die als $\bar{U}$-Dominanzregel bezeichnet wird, vergleicht die Ressourcenbeschränkungsmatrizen von zwei voneinander nicht erreichbaren Knoten im Enumerationsbaum. Seien $\bar{U}'$ und $\bar{U}''$ die Ressourcenbeschränkungsmatrizen von zwei voneinander nicht erreichbaren Knoten W' und W'' im Enumerationsbaum, für die $\bar{U}' \leq \bar{U}''$ gilt. Dann dominiert W'' den Enumerationsknoten W'. Dieser Zusammenhang kann wie folgt hergeleitet werden. Zunächst sei ein beliebiger Enumerationsknoten W mit der Ressourcenbeschränkungsmatrix $\bar{U}$ gegeben. Dann gelten $W \subseteq W(\bar{U})$ und $\mathcal{S}(W) = \mathcal{S}(W(\bar{U}))$, da keiner der betrachteten Konsistenztests aus Abschnitt 4.2 einen zulässigen Schedule aus dem Suchraum $\mathcal{S}_T(W)$ ausschließt. Da weiterhin $\mathcal{S}_T(W(\bar{U}')) \subseteq \mathcal{S}_T(W(\bar{U}''))$

aus $\bar{U}' \leq \bar{U}''$ abgeleitet werden kann, sodass auch $\mathcal{S}(W(\bar{U}')) \subseteq \mathcal{S}(W(\bar{U}''))$ gilt, folgt schließlich $\mathcal{S}(W') \subseteq \mathcal{S}(W'')$.

Im Gegensatz zur $\bar{U}$-Dominanzregel ist es für die zweite Dominanzregel nicht erforderlich, zusätzliche Informationen in den Enumerationsknoten abzuspeichern. Stattdessen werden die Startzeitbeschränkungen W' und W'' von zwei voneinander nicht erreichbaren Knoten im Enumerationsbaum direkt miteinander verglichen. Die Redundanz eines Teils des Enumerationsbaums wird durch die sogenannte W-Dominanzregel durch die Bedingung $W' \subseteq W''$ überprüft, wobei die Dominanz von Knoten W'' direkt aus $\mathcal{S}_T(W') \subseteq \mathcal{S}_T(W'')$ folgt.

Im Folgenden wird die Effektivität beider Dominanzregeln einander gegenübergestellt, wobei unter der Effektivität die Anzahl der detektierten redundanten Knoten im Enumerationsverlauf verstanden wird. Zunächst ist zu beachten, dass die Effektivität der W-Dominanzregel im Gegensatz zur $\bar{U}$-Dominanzregel von der Verwendung der Konsistenztests im Enumerationsverlauf abhängt. Es kann gezeigt werden, dass unter bestimmten Voraussetzungen im Hinblick auf die Konsistenztests die W-Dominanzregel die $\bar{U}$-Dominanzregel in dem Sinne dominiert, dass jeder Knoten, den die $\bar{U}$-Dominanzregel als redundant detektiert, auch durch die W-Dominanzregel als redundant identifiziert wird, d. h., $\bar{U}' \leq \bar{U}''$ impliziert $W' \subseteq W''$. Zunächst wird angenommen, dass keine Konsistenztests im Enumerationsverlauf verwendet werden. Da in diesem Fall die Zusammenhänge $W' = W(\bar{U}')$ und $W'' = W(\bar{U}'')$ gelten, ergibt sich die Dominanz der W-Dominanzregel direkt. Als Nächstes wird vorausgesetzt, dass in jedem Enumerationsknoten der Fixpunkt einer beliebigen Menge Γ von Konsistenztests aus Abschnitt 4.2 bestimmt wird. Aufgrund der Monotonie aller Konsistenztests folgt aus $\bar{U}' \leq \bar{U}''$ bzw. $W(\bar{U}') \subseteq W(\bar{U}'')$ schließlich $W' \subseteq W''$. Zusammenfassend kann eine Dominanz der W-Dominanzregel über die $\bar{U}$-Dominanzregel abgeleitet werden, falls keine Konsistenztests verwendet werden oder in jedem Enumerationsknoten ein Fixpunkt einer beliebigen Menge von Konsistenztests Γ bestimmt wird. Es ist dabei zu beachten, dass die geschilderten Zusammenhänge zwischen den Dominanzregeln durch die Ausführung eines Preprocessing-Verfahrens im Wurzelknoten nicht beeinflusst werden.

Abschließend werden die Zeitkomplexitäten beider Dominanzregeln zur Überprüfung der jeweiligen Bedingung für zwei voneinander nicht erreichbare Knoten W' und W'' im Enumerationsbaum betrachtet. Für die $\bar{U}$-Dominanzregel kann eine Zeitkomplexität von $\mathcal{O}(|V||\mathcal{R}|)$ und für die W-Dominanzregel eine Zeitkomplexität von $\mathcal{O}(|V| + \max(\mathcal{B}', \mathcal{B}''))$ mit $\mathcal{B}'$ und $\mathcal{B}''$ als Anzahl der Startzeitunterbrechungen in W' und W'' abgeleitet werden.

Auf die Implementierung der Dominanzregeln im Branch-and-Bound-Verfahren wird in Abschnitt 5.1.4 eingegangen.

5.1.3 Partitionierung des Suchraums

Die Dominanzregeln, die im vorangegangenen Abschnitt behandelt wurden, sind im Allgemeinen nicht in der Lage, Redundanzen im Enumerationsbaum vollständig auszuschließen bzw. zu verhindern, dass Teilmengen des gesamten Suchraums $\mathcal{S}_T$ mehrfach untersucht werden. Im Folgenden wird gezeigt, wie der Suchraum jedes Enumerationsknotens partitioniert werden kann, um sicherzustellen, dass jede beliebige Teilmenge des gesamten Suchraums $\mathcal{S}_T$ nicht mehr als ein Mal im Enumerationsverlauf untersucht wird. Es ist dabei zu beachten, dass ein ähnlicher Ansatz bereits in Murty (1968) für das Zuordnungsproblem eingesetzt wurde. Um den Suchraum $\mathcal{S}_T(W)$ jedes Enumerationsknotens im Verzweigungsschritt zu partitionieren, wird das Enumerationsschema in Algorithmus 5.1 wie folgt angepasst. Sei W die Startzeitbeschränkung eines beliebigen Enumerationsknotens mit $S = \min \mathcal{S}_T(W) \notin \mathcal{S}_R$ und sei $k \in \mathcal{R}^c(S)$ die gewählte Konfliktressource für die Zerlegung des Suchraums $\mathcal{S}_T(W)$. Weiterhin sei durch $(i_1, i_2, \ldots, i_\mu, \ldots, i_{|V_k(S)|})$ mit $V_k(S) := \{i \in V_k \,|\, r^u_{ik}(S_i) > 0\}$ die Reihenfolge aller Vorgänge gegeben, in der die zugehörigen Nachfolgerknoten, wie in Abschnitt 5.1.1 beschrieben, generiert werden. Dann wird für den Nachfolgerknoten W^{i_μ} mit $\mu \in \{1, 2, \ldots, |V_k(S)|\}$, der durch die Verzweigung über Vorgang $i_\mu \in V_k(S)$ generiert wird, $W_i^{i_\mu} := W_i$ für alle $i \in V \setminus V_k(S)$ und

$$W_{i_s}^{i_\mu} := \begin{cases} W_{i_s}, & \text{falls } s < \mu \\ \{\tau \in W_{i_s} \,|\, r^u_{i_s,k}(\tau) < r^u_{i_s,k}(S)\}, & \text{falls } s = \mu \\ \{\tau \in W_{i_s} \,|\, r^u_{i_s,k}(\tau) \geq r^u_{i_s,k}(S)\}, & \text{sonst} \end{cases}$$

für alle $s \in \{1, 2, \ldots, |V_k(S)|\}$ gesetzt.

Im Folgenden wird gezeigt, dass die angepasste Verzweigung zu einer Partitionierung des Suchraums $\mathcal{S}_T(W)$ führt, sodass die Bedingung $\mathcal{S}_T(W) \cap \mathcal{S} = \bigcup_{i \in V_k(S)} (\mathcal{S}_T(W^i) \cap \mathcal{S})$ erfüllt ist, wodurch die Korrektheit des Enumerationsschemas weiterhin gilt. Zunächst folgt $\mathcal{S}_T(W^{i_{\mu'}}) \cap \mathcal{S}_T(W^{i_{\mu''}}) = \emptyset$ für alle $\mu', \mu'' \in \{1, 2, \ldots, |V_k(S)|\}$ mit $\mu' \neq \mu''$ direkt aus der Berechnungsvorschrift der angepassten Verzweigung, sodass eine Partitionierung von $\mathcal{S}_T(W)$ gegeben ist. Entsprechend verbleibt zu zeigen, dass jeder zulässige Schedule in $\mathcal{S}_T(W)$ im Suchraum eines direkten Nachfolgerknotens liegt. Dafür ist es ausreichend zu

zeigen, dass die Bedingung

$$\bigcup_{\mu=1}^{|V_k(S)|} \mathcal{S}_T(\overline{W}^{i_\mu}) = \bigcup_{\mu=1}^{|V_k(S)|} \mathcal{S}_T(W^{i_\mu}) \tag{5.1}$$

mit $\overline{W}^{i_\mu}$ als Startzeitbeschränkung, die durch den Verzweigungsschritt in Algorithmus 5.1 für den Vorgang $i_\mu \in V_k(S)$ bestimmt wird, gilt. Die Korrektheit der Gleichung (5.1) folgt schließlich aus $\mathcal{S}_T(W^{i_{|V_k(S)|}}) = \mathcal{S}_T(\overline{W}^{i_{|V_k(S)|}})$, $\mathcal{S}_T(\overline{W}^{i_\mu}) \setminus \mathcal{S}_T(W^{i_\mu}) = \bigcup_{\mu'=\mu+1}^{|V_k(S)|} \mathcal{S}_T(W^{i_{\mu'}})$ für alle $\mu = \{1, 2, \ldots, |V_k(S)| - 1\}$ und $\mathcal{S}_T(W^i) \subseteq \mathcal{S}_T(\overline{W}^i)$ für alle $i \in V_k(S)$.

5.1.4 Branch-and-Bound-Algorithmus

Im Folgenden wird ein Branch-and-Bound-Verfahren, das auf dem relaxationsbasierten Enumerationsschema aus Abschnitt 5.1.1 basiert, für das RCPSP/max-π entwickelt. Im ersten Teil dieses Abschnitts wird zunächst die Suchstrategie des Branch-and-Bound-Verfahrens zur Festlegung der schrittweisen Generierung des Enumerationsbaums betrachtet. Anschließend wird das relaxationsbasierte Enumerationsschema in Algorithmus 5.1 zu einem Branch-and-Bound-Verfahren erweitert, indem untere Schrankenwerte, Konsistenztests und Dominanzregeln hinzugefügt werden.

Im ersten Schritt wird die Suchstrategie des relaxationsbasierten Branch-and-Bound-Verfahrens vorgestellt. Dazu wird die Suchstrategie in vier verschiedene Strategien aufgeteilt, die im Folgenden als Traversierungs-, Generierungs-, Reihenfolge- und Verzweigungsstrategie bezeichnet werden. Die Traversierungsstrategie legt zunächst die Reihenfolge fest, in der alle Enumerationsknoten, die bislang generiert, aber noch nicht vollständig untersucht wurden, betrachtet werden. Ein Knoten wird dabei als vollständig untersucht bezeichnet, falls entweder alle seine direkten Nachfolgerknoten bereits generiert wurden oder gezeigt werden konnte, dass im Suchraum des Knotens kein zulässiger Schedule mit einer kürzeren Projektdauer als der einer bereits bekannten Lösung existiert. Eine Traversierungsstrategie, die für das relaxationsbasierte Branch-and-Bound-Verfahren verwendet wird, ist die sogenannte Tiefensuche (DFS) (engl. Depth-First Search), für die alle generierten Knoten nach der LIFO-Regel (engl. Last-In-First-Out) durchlaufen werden. Da experimentelle Performance-Analysen zeigen konnten, dass die DFS häufig erst nach einer längeren Laufzeit eine erste zulässige Lösung bestimmt, wurde eine weitere Traversierungsstrategie aus Watermeyer und Zimmermann (2020), die sogenannte Scattered-Path-Suche (SPS), implementiert. Die grundlegende Idee der SPS basiert auf einer Streuung der Suche im Enumerationsbaum, wobei insbesondere der Nachteil der

DFS vermieden wird, dass ein Bereich des Suchbaums, der keine oder nur schlechte Lösungen enthält, nicht mehr verlassen werden kann. Die SPS unterscheidet sich von der DFS lediglich dadurch, dass sie nach Ablauf einer vorgegebenen Zeitspanne unter allen noch nicht vollständig untersuchten Knoten auf der kleinsten Ebene des Suchbaums einen Knoten mit kleinstem unterem Schrankenwert für die kürzeste Projektdauer als nächsten zu untersuchenden Knoten auswählt. Eine weitere Variante der SPS betrachtet zusätzlich Prioritätsregelwerte, falls die unteren Schrankenwerte mehrerer infrage kommender Knoten gleich sind, wobei die Prioritätsregelwerte wie für die nachfolgend beschriebene Reihenfolgestrategie bestimmt werden. Die entsprechende Variante der SPS wird im weiteren Verlauf dieser Arbeit durch SPS$^+$ gekennzeichnet.

Es ist zu beachten, dass die Traversierungsstrategie weder festlegt, wie viele direkte Nachfolgerknoten in einem Verzweigungsschritt generiert werden, noch die Reihenfolge vorgibt, in der die generierten Nachfolgerknoten im weiteren Enumerationsverlauf untersucht werden. Stattdessen werden diese Spezifikationen des Branch-and-Bound-Verfahrens durch die Generierungs- und Reihenfolgestrategie festgelegt. Für die Generierungsstrategie werden zunächst zwei Alternativen unterschieden, bei denen entweder alle direkten Nachfolger eines Enumerationsknotens im Verzweigungsschritt generiert werden (all) oder nur eine vorgegebene maximale Anzahl (restr). Dabei ist zu beachten, dass jeder Knoten im Enumerationsverlauf solange untersucht wird, bis entweder alle seine direkten Nachfolger generiert wurden oder gezeigt werden konnte, dass der zugehörige Suchraum leer ist. Neben der Anzahl der generierten direkten Nachfolgerknoten in jedem Verzweigungsschritt konnte durch experimentelle Voruntersuchungen gezeigt werden, dass die Reihenfolge, in der alle generierten Knoten im weiteren Enumerationsverlauf untersucht werden (Reihenfolgestrategie), ebenfalls einen entscheidenden Einfluss auf die Performance des Branch-and-Bound-Verfahrens hat. Wie bereits für das RCPSP/max festgestellt werden konnte (vgl. z. B. De Reyck und Herroelen, 1998), ist es für das RCPSP/max-π i. d. R. auch von Vorteil, alle generierten Nachfolgerknoten in einer Reihenfolge nach nichtfallenden Werten der unteren Schranken für die kürzeste Projektdauer im weiteren Enumerationsverlauf zu untersuchen. Da die Schrankenwerte für die kürzeste Projektdauer unter den direkten Nachfolgern eines Enumerationsknotens häufig gleich sind, betrachtet die Reihenfolgestrategie zusätzlich Prioritätsregelwerte, um die Güte möglicher zulässiger Lösungen im Suchraum eines Nachfolgerknotens abzuschätzen. Im Folgenden werden Priortitätsregeln vorgestellt, die in experimentellen Voruntersuchungen die besten Ergebnisse zeigen konnten. Dazu sei ein Knoten W im Enumerationsverlauf von Algorithmus 5.1 mit einem Schedule $S = \min \mathcal{S}_T(W) \notin \mathcal{S}_R$ gegeben, für den eine Konfliktressource $k \in \mathcal{R}^c(S)$ im Verzweigungsschritt ausgewählt wurde. Weiterhin sei angenommen, dass $(i_1, i_2, \ldots, i_s)$ der Reihenfolge aller generierten direkten

Nachfolgerknoten entspricht, die nach nichtfallenden Werten der unteren Schranken der kürzesten Projektdauer sortiert sind, mit $i_\mu \in V_k(S) := \{i \in V \mid r^u_{ik}(S_i) > 0\}$ für alle $\mu \in \{1, 2, \ldots, s\}$ und $s \leq |V_k(S)|$, sodass $i_{\mu'}$ vor $i_{\mu''}$ untersucht wird, falls $\mu' < \mu''$ gilt. Dann wird die Reihenfolge, in der alle direkten Nachfolgerknoten mit gleichem Schrankenwert der kürzesten Projektdauer in der Liste $(i_1, i_2, \ldots, i_s)$ untersucht werden, zusätzlich nach Prioritätsregelwerten in Abhängigkeit einer gewählten Prioritätsregel sortiert, wobei die Vorgänge entweder nach nichtfallenden (min) oder nichtsteigenden (max) Prioritätsregelwerten sortiert werden. Für die ersten beiden Regeln wird jedem Vorgang $i \in V_k(S)$ ein Prioritätsregelwert π_i in Abhängigkeit von der Belegung der Konfliktressource k durch $\pi_i = r^u_{ik}(S_i)$ für die RU-Regel (engl. Resource Usage) und $\pi_i = r^u_{ik}(S_i)/p_i$ für die RUT-Regel (engl. Resource Usage per Time) zugeordnet. Die DST-Regel (engl. Delayed Start Time) berücksichtigt die minimale Erhöhung des aktuellen Startzeitpunkts S_i von Vorgang $i \in V_k(S)$ durch den Verzweigungsschritt bzw. durch die Einschränkung der Belegung der Konfliktressource k durch Vorgang i, indem $\pi_i = \Delta_{ik}$ mit

$$\Delta_{ik} := \begin{cases} \min\{\tau \in W^{\Delta}_{ik} \mid \tau \geq S_i\} - S_i, & \text{falls } \{\tau \in W^{\Delta}_{ik} \mid \tau \geq S_i\} \neq \emptyset \\ \max W_i - S_i + 1, & \text{sonst} \end{cases}$$

und $W^{\Delta}_{ik} := \{\tau \in W_i \mid r^u_{ik}(\tau) < r^u_{ik}(S_i)\}$ gesetzt wird. Im Gegensatz zu den bisherigen Prioritätsregeln, die von der gewählten Konfliktressource $k \in \mathcal{R}^c(S)$ abhängen, basieren die folgenden Regeln auf Pufferzeiten. Die erste Prioritätsregel TF (engl. Total Float) bestimmt die sogenannte Gesamtpufferzeit TF_i für jeden Vorgang $i \in V_k(S)$, die der maximalen Erhöhung des Startzeitpunkts S_i entspricht, sodass im Suchraum $\mathcal{S}_T(W)$ weiterhin mindestens ein W-zulässiger Schedule mit einer kürzeren Projektdauer als die einer bislang besten bekannten Lösung existiert. Entsprechend ist der Prioritätsregelwert π_i von Vorgang $i \in V_k(S)$ durch

$$TF_i := LS^{UB}_i(W) - S_i$$

gegeben, mit $LS^{UB}_i(W) := LS_i(W, n+1, UB-1)$ und UB als Projektdauer der bislang besten bekannten zulässigen Lösung, soweit bereits eine zulässige Lösung bestimmt wurde, oder andernfalls $UB := \bar{d} + 1$. Die EFF-Regel (engl. Early Free Float) bestimmt für jeden Vorgang $i \in V_k(S)$ die freie Pufferzeit EFF_i, die der maximalen Erhöhung des Startzeitpunkts S_i entspricht, sodass alle anderen Vorgänge des Projekts weiterhin zum frühestmöglichen W-zulässigen Startzeitpunkt beginnen können. Entsprechend gibt

$$EFF_i := \max\{\tau \in W_i \mid \tau \leq \min_{j \in Succ(i)} (S_j - \delta_{ij})\} - S_i$$

den Prioritätsregelwert π_i von Vorgang $i \in V_k(S)$ an. Für die Prioritätsregeln DST, TF und EFF wurde jeweils eine weitere Variante implementiert, die die Anzahl der Zeitpunkte in der Startzeitbeschränkung berücksichtigt, die durch die Erhöhung des Startzeitpunkts des betrachteten Vorgangs übersprungen werden. Die entsprechenden Prioritätsregeln DST^{I}, TF^{I} und EFF^{I} weisen jedem Vorgang $i \in V_k(S)$ einen Prioritätsregelwert $\pi_i = |]S_i, S_i + \Delta_i] \cap W_i|$ mit $\Delta_i \in \{TF_i, EFF_i, \Delta_{ik}\}$ zu. Die Prioritätsregeln, die bislang für die Festlegung einer Reihenfolge für die Untersuchung aller generierten direkten Nachfolger eines Knotens im Enumerationsverlauf eingeführt wurden, können im Branch-and-Bound-Verfahren ebenfalls zur Bestimmung einer Reihenfolge für die Generierung der direkten Nachfolgerknoten eingesetzt werden. Für diese Variante der Generierungsstrategie (restrCL) wird eine Liste aller direkten Nachfolgerknoten erstellt und in Abhängigkeit der verwendeten Reihenfolgestrategie sortiert. Es ist dabei zu beachten, dass ohne Vorgabe einer Reihenfolge für die Generierung der direkten Nachfolgerknoten durch Prioritätsregelwerte (all, restr) die direkten Nachfolgerknoten in einer Reihenfolge nach fallenden Indizes der zugehörigen Vorgänge generiert werden.

Den letzten Teil der Suchstrategie stellt schließlich die sogenannte Verzweigungsstrategie dar, die vorgibt, nach welcher Regel die Konfliktressource $k \in \mathcal{R}^c(S)$ in einem Verzweigungsschritt ausgewählt wird. Wie für die Bestimmung der Reihenfolgen zur Generierung und Untersuchung der Enumerationsknoten werden für die Auswahl der Konfliktressource k ebenfalls Prioritätsregeln verwendet. Basierend auf den Prioritätsregeln, die bereits für die Reihenfolgestrategie eingeführt wurden, können Prioritätsregeln für die Verzweigungsstrategie durch die Zuordnung der Prioritätsregelwerte $\pi_k = \sum_{i \in V_k(S)} \pi_i / |V_k(S)|$ zu jeder Konfliktressource $k \in \mathcal{R}^c(S)$ direkt abgeleitet werden. Als Beispiel wird die Prioritätsregel TF betrachtet, die jeder Konfliktressource $k \in \mathcal{R}^c(S)$ einen Prioritätsregelwert $\pi_k = \sum_{i \in V_k(S)} TF_i / |V_k(S)|$ zuordnet, sodass π_k dem Durchschnittswert der Gesamtpufferzeiten über alle Vorgänge $i \in V_k(S)$ entspricht. Zur Vereinfachung werden im weiteren Verlauf die Bezeichner für die Prioritätsregeln der Reihenfolgestrategie für die Verzweigungsstrategie unverändert übernommen. Neben den Prioritätsregeln, die sich direkt aus der Reihenfolgestrategie ergeben, wurden zusätzlich weitere Prioritätsregeln für die Verzweigungsstrategie untersucht. Die ACO-Regel (engl. Absolute Capacity Overrun) verwendet dabei als Prioritätsregelwert π_k die Kapazitätsüberschreitung $\Delta_k := r_k^c(S) - R_k$, die RCO-Regel (engl. Relative Capacity Overrun) die relative Kapazitätsüberschreitung Δ_k / R_k und die NCA-Regel (engl. Number of Consuming Activities) die Anzahl der Vorgänge mit einer positiven Ressourceninanspruchnahme $|V_k(S)|$. Basierend auf der Prioritätsregel wird die Konfliktressource schließlich durch

$$k := \min\{k' \in \mathcal{R}^c(S) \mid \pi_{k'} = \operatorname*{ext}_{l \in \mathcal{R}^c(S)} \pi_l\}$$

bestimmt, wobei ext $\in \{\min, \max\}$ festlegt, ob kleinere (min) oder größere (max) Prioritätsregelwerte vorgezogen werden.

Algorithmus 5.2 beschreibt den relaxationsbasierten Branch-and-Bound-Algorithmus, wobei sich die Darstellung des Verfahrens auf eine Tiefensuche und eine Generierung aller direkten Nachfolgerknoten in jedem Verzweigungsschritt beschränkt. Entsprechend werden die weiteren Alternativen, die für die Traversierungs- und die Generierungsstrategie vorgestellt wurden, in Algorithmus 5.2 nicht abgebildet.

Zu Beginn von Algorithmus 5.2 wird zunächst die Startzeitbeschränkung W initialisiert, wobei der Floyd-Warshall-Algorithmus mit einer Zeikomplexität von $\mathcal{O}(|V|^3)$ (vgl. Ahuja et al., 1993, Abschnitt 5.6) ausgeführt wird, um entweder die Distanzmatrix D zu bestimmen oder zu zeigen, dass ein Zyklus positiver Länge in Projektnetzplan N vorliegt ($\mathcal{S} = \emptyset$). Anschließend wird in einem Preprocessing-Schritt, der auf der Startzeitbeschränkung W ausgeführt wird, der eindeutige Fixpunkt der Konsistenztests aus der Menge Γ^{W} unter Berücksichtigung aller Ressourcen bestimmt, d. h., es wird $W := \gamma_W^{\infty}[\mathcal{R}](W)$ gesetzt (s. Abschnitt 4.2). Im Fall, dass nach der Ausführung des Preprocessing-Schritts die Existenz eines zulässigen Schedules nicht ausgeschlossen werden kann ($\mathcal{S}_T(W) \neq \emptyset$), wird die untere Schranke für die kürzeste Projektdauer über alle zulässigen Schedules auf $LB := S_{n+1}$ gesetzt ($S_{n+1} = LB0^{\pi}$). Dabei ist zu beachten, dass durch die Berechnung der destruktiven unteren Schranke LBD^{π} mit $LB^{start} := LB0^{\pi}$ und $UB^{start} := \bar{d}$ aufgrund des Preprocessings kein besserer Schrankenwert erzielt werden kann. Abschließend wird der Wurzelknoten (W, S, LB) auf den Stack Ω gelegt und die obere Schranke der Projektdauer durch $UB := \bar{d} + 1$ initialisiert.

In jeder Iteration wird der oberste Knoten (W, S, LB) vom Stack Ω entfernt. Falls der Knoten (W, S, LB) nicht ausgelotet werden kann, da die Bedingung $LB < UB$ erfüllt ist, werden Konsistenztests auf der Startzeitbeschränkung W ausgeführt. Im weiteren Verlauf dieser Arbeit ist zu beachten, dass für den T- und TB-Konsistenztest bei der Ausführung innerhalb eines Branch-and-Bound-Verfahrens zusätzlich die Bedingung $S_{n+1} < UB$ berücksichtigt wird, sodass die Projektdauer einer bereits ermittelten zulässigen Lösung mit einbezogen wird, um die Effektivität der Konsistenztests zu steigern. Für den Fall, dass nach der Ausführung der Konsistenztests kein W-zulässiger Schedule mehr im Suchraum des Knotens (W, S, LB) mit einer geringeren Projektdauer als UB vorliegt, d. h., falls $\mathcal{S}_T^{UB}(W) := \widehat{\mathcal{S}}_T(W, n+1, UB-1) = \emptyset$ gilt, wird der Knoten ausgelotet. Andernfalls wird der Schedule S durch $S := \min \mathcal{S}_T(W)$ aktualisiert, soweit mindestens ein Zeitpunkt aus der Startzeitbeschränkung W durch die Konsistenztests entfernt wurde. Soweit ein ressourcenzulässiger Schedule S vorliegt, wird S als neue beste zulässige Lösung durch $S^* := S$ abgespeichert und $UB := S_{n+1}^*$ gesetzt. Gilt dagegen $S \notin \mathcal{S}_R$, wird der Suchraum

Algorithmus 5.2: Relaxationsbasierter Branch-and-Bound-Algorithmus

Input: RCPSP/max-π-Instanz
Output: Optimaler Schedule S^*

Bestimme die Distanzmatrix $D = (d_{ij})_{i,j \in V}$
if $\mathcal{S}_T = \emptyset$ **then** terminate ($\mathcal{S} = \emptyset$)
Setze $ES_i := d_{0i}$, $LS_i := -d_{i0}$ und $W_i := [ES_i, LS_i] \cap \mathbb{Z}$ für alle $i \in V$
Führe Preprocessing-Schritt auf W aus
if $\mathcal{S}_T(W) = \emptyset$ **then** terminate ($\mathcal{S} = \emptyset$)
$S := \min \mathcal{S}_T(W), \quad LB := S_{n+1}$
$\Omega := \{(W, S, LB)\}, \quad UB := \bar{d} + 1$
while $\Omega \neq \emptyset$ **do**
  Entferne den obersten Knoten (W, S, LB) vom Stack Ω
  if $LB < UB$ **then**
    Führe Konsistenztests auf W aus
    if $\mathcal{S}_T^{UB}(W) \neq \emptyset$ **then**
      Aktualisiere $S := \min \mathcal{S}_T(W)$
      if $S \in \mathcal{S}_R$ **then**
        $S^* := S, \quad UB := S^*_{n+1}$
      else
        Initialisiere $\Lambda := \emptyset$ und wähle eine Konfliktressource $k \in \mathcal{R}^c(S)$ gemäß der Verzweigungsstrategie
        forall $i \in \{j \in V_k \mid r^u_{jk}(S_j) > 0\}$ **do**
          $W' := W, \qquad W'_i := W'_i \cap W_{ik}(r^u_{ik}(S_i) - 1)$
          if $\mathcal{S}_T^{UB}(W') \neq \emptyset$ **then**
            Aktualisiere $S' := \min \mathcal{S}_T(W')$
            if $S' \in \mathcal{S}_R$ **then**
              $S^* := S', \quad UB := S^*_{n+1}$
            else
              Wende Dominanzregeln an und berechne die untere Schranke LB'
              if W' wird nicht dominiert und $LB' < UB$ **then**
                $\Lambda := \Lambda \cup \{(W', S', LB')\}$
        Entferne alle Knoten aus der Liste Λ und lege sie gemäß der Reihenfolgestrategie auf den Stack Ω
if $UB = \bar{d} + 1$ **then** terminate ($\mathcal{S} = \emptyset$)
else return S^*

$\mathcal{S}_T(W)$ in einem Verzweigungsschritt, wie in Abschnitt 5.1.1 beschrieben, zerlegt, wobei die Konfliktressource $k \in \mathcal{R}^c(S)$ gemäß der Verzweigungsstrategie ausgewählt wird. Wie in Abschnitt 5.1.3 gezeigt wurde, kann der Verzweigungsschritt auch durch eine Partitionierung des Suchraums ersetzt werden. Für jeden generierten direkten Nachfolgerknoten, der mindestens einen W-zulässigen Schedule mit einer geringeren Projektdauer als UB

enthält, wird der Schedule $S' := \min \mathcal{S}_T(W')$ bestimmt. Falls S' ressourcenzulässig ist, wird S' als neue beste zulässige Lösung abgespeichert und UB entsprechend aktualisiert. Andernfalls wird mit Hilfe von Dominanzregeln aus Abschnitt 5.1.2 überprüft, ob der direkte Nachfolgerknoten redundant ist und somit ausgelotet werden kann. Für die Anwendung der Dominanzregeln wird der direkte Nachfolgerknoten allen Knoten aus der Menge Ω gegenübergestellt, die keine Vorgängerknoten im Suchbaum darstellen. Es ist dabei zu beachten, dass, falls im Verzweigungsschritt nicht alle direkten Nachfolgerknoten auf einmal generiert werden (restr, restrCL), alle Vorgänger eines durch die Dominanzregeln bereits ausgeloteten (redundanten) Knotens im weiteren Enumerationsverlauf von den Anwendungen der Dominanzregeln ausgenommen werden, wodurch die Korrektheit des Verfahrens sichergestellt wird. Falls nicht gezeigt werden kann, dass der betrachtete Nachfolgerknoten redundant ist, wird die untere Schranke LB' berechnet, die entweder der unteren Schranke $LB0^{\pi} = S'_{n+1}$ oder der destruktiven unteren Schranke LBD^{π} entspricht, wobei für die Berechnung von LBD^{π} die Werte $LB^{start} := LB0^{\pi}$ und $UB^{start} := UB - 1$ in Algorithmus 4.14 gesetzt werden (s. Abschnitt 4.3). Gilt $LB' < UB$, wird der direkte Nachfolgerknoten schließlich der Liste Λ hinzugefügt. Nach der Generierung aller direkten Nachfolger werden alle Knoten aus der Liste Λ entfernt und gemäß der Reihenfolgestrategie auf den Stack Ω gelegt, sodass der oberste Knoten unter allen generierten direkten Nachfolgerknoten den kleinsten unteren Schrankenwert LB' aufweist. Das beschriebene Verfahren wird solange ausgeführt, bis alle Enumerationsknoten vollständig untersucht wurden ($\Omega = \emptyset$). Nach Ablauf des relaxationsbasierten Branch-and-Bound-Verfahrens wird schließlich entweder ein optimaler Schedule S^* zurückgegeben oder im Fall $UB = \bar{d}+1$ die Unzulässigkeit der betrachteten Probleminstanz gezeigt.

5.2 Ein partitionsbasierter Ansatz

In Abschnitt 5.2.1 wird das Enumerationsschema für das Branch-and-Bound-Verfahren in Abschnitt 5.2.2 vorgestellt, das der Ressourcenrelaxation von Problem (P) in jedem Enumerationsschritt sowohl eine obere als auch eine untere Schranke für die Belegung einer Ressource durch einen Vorgang hinzufügt, wodurch die Mengen aller zeitzulässigen Startzeitpunkte der Vorgänge des Projekts schrittweise partitioniert werden. Abschließend wird in Abschnitt 5.2.2 das Branch-and-Bound-Verfahren beschrieben, das auf dem Enumerationsansatz in Abschnitt 5.2.1 basiert.

5.2.1 Enumerationsschema

In diesem Abschnitt wird ein weiteres relaxationsbasiertes Enumerationsschema für das RCPSP/max-π entwickelt, das der Ressourcenrelaxation von Problem (P) in jedem Enumerationsschritt solange eine obere als auch eine untere Schranke für die Belegung einer Ressource durch einen Vorgang des Projekts hinzufügt, bis entweder die optimale Lösung eines Enumerationsknotens zulässig ist oder der zugehörige Suchraum leer ist. Da aus der Enumeration eine schrittweise Partitionierung der Mengen aller zeitzulässigen Startzeitpunkte der Vorgänge folgt, wird zur Abgrenzung vom Enumerationsansatz in Abschnitt 5.1.1 das in diesem Abschnitt behandelte Enumerationsschema im weiteren Verlauf dieser Arbeit als partitionsbasiert bezeichnet, wobei zu beachten ist, dass der Ansatz ebenfalls zur Kategorie der relaxationsbasierten Verfahren zählt. Der partitionsbasierte Ansatz unterscheidet sich vom Enumerationsverfahren in Abschitt 5.1.1 und von anderen klassischen relaxationsbasierten Ansätzen (s. z. B. De Reyck und Herroelen, 1998; Fest et al., 1999) insbesondere dadurch, dass die optimale Lösung eines Knotens im Enumerationsverlauf durch die Zerlegung des Suchraums nicht von den weiteren Betrachtungen ausgeschlossen wird. Die Vorteile, die sich daraus ergeben, sind unter anderem eine konstante maximale Anzahl an direkten Nachfolgerknoten in jedem Verzweigungsschritt sowie eine redundanzfreie Enumeration.

Der Ablauf des partitionsbasierten Enumerationsschemas wird in Algorithmus 5.3 beschrieben. Das Verfahren unterscheidet sich vom relaxationsbasierten Ansatz in Algorithmus 5.1 lediglich durch die Zerlegung des Suchraums eines Enumerationsknotens, sodass im Folgenden nur noch der Verzweigungsschritt in Algorithmus 5.3 betrachtet wird. Analog zum relaxationsbasierten Ansatz wird jeder Knoten im Enumerationsbaum durch eine Startzeitbeschränkung W repräsentiert, wobei der Wurzelknoten durch $W_i := [ES_i, LS_i] \cap \mathbb{Z}$ für alle Vorgänge $i \in V$ initialisiert wird, sodass zu Beginn der Enumeration jedem Vorgang alle zeitzulässigen Startzeitpunkte zugeordnet werden. Sobald im Enumerationsverlauf ein Knoten mit $S = \min \mathcal{S}_T(W) \notin \mathcal{S}_R$ vorliegt, wird zunächst überprüft, ob die Existenz eines zulässigen Schedules im Suchraum $\mathcal{S}_T(W)$ bereits ausgeschlossen werden kann ($\mathcal{S}(W) = \emptyset$). Dazu wird die Mindestinanspruchnahme $\underline{r}_k^c(W)$ jeder Konfliktressource $k \in \mathcal{R}^c(S)$ bestimmt, wobei für jeden Vorgang $i \in V$ alle Zeitpunkte in W_i zwischen dem frühesten und dem spätesten W-zulässigen Startzeitpunkt betrachtet werden. Die Mindestinanspruchnahme $\underline{r}_k^c(W)$ ergibt sich entsprechend aus der Summe der Mindestinanspruchnahmen $\underline{r}_{ik}^c(W) := \min\{r_{ik}^c(\tau) \mid \tau \in W_i \cap [ES_i(W), LS_i(W)]\}$ aller Vorgänge $i \in V_k$. Falls $\underline{r}_k^c(W) > R_k$ für mindestens eine Konfliktressource $k \in \mathcal{R}^c(S)$ gilt, folgt $\mathcal{S}(W) = \emptyset$, sodass der Suchraum $\mathcal{S}_T(W)$ nicht weiter zerlegt wird und Knoten W ein Blatt des Enumerationsbaums darstellt. Kann dagegen ein zulässiger Sche-

Algorithmus 5.3: Partitionsbasiertes Enumerationsschema

Input: RCPSP/max-π-Instanz
Output: Menge Φ aller generierten Schedules

```
1  Bestimme die Schedules ES und LS
2  if S_T = ∅ then terminate (S = ∅)
3  W_i := [ES_i, LS_i] ∩ ℤ für alle i ∈ V
4  Ω := {W},        Φ := ∅
5  while Ω ≠ ∅ do
6  |  Entferne W aus der Menge Ω
7  |  S := min S_T(W)
8  |  if S ∈ S_R then
9  |  |  Φ := Φ ∪ {S}
10 |  else if ∄k ∈ R : r_k^c(W) > R_k then
11 |  |  Wähle ein Paar (k, i) ∈ R^c(S) × V_k mit r_ik^u(S_i) > r_ik^u(W)
12 |  |  Berechne ū_ik := ⌊(r_ik^u(S_i) + r_ik^u(W)) / 2⌋
13 |  |  W' := W,        W_i' := {τ ∈ W_i | r_ik^u(τ) > ū_ik}
14 |  |  Ω := Ω ∪ {W'}
15 |  |  W'' := W,       W_i'' := {τ ∈ W_i | r_ik^u(τ) ≤ ū_ik}
16 |  |  if S_T(W'') ≠ ∅ then
17 |  |  |  Ω := Ω ∪ {W''}
18 return Φ
```

dule $S \in \mathcal{S}(W)$ nicht ausgeschlossen werden, wird der Suchraum $\mathcal{S}_T(W)$ zerlegt. Dazu wird zunächst ein Paar $(k,i) \in \mathcal{R}^c(S) \times V_k$ mit $r^u_{ik}(S_i) > \underline{r}^u_{ik}(W)$ ausgewählt, wobei $\underline{r}^u_{ik}(W) := \min\{r^u_{ik}(\tau) \mid \tau \in W_i \cap [ES_i(W), LS_i(W)]\}$ der Mindestbelegung der Ressource k durch Vorgang i entspricht. Es ist dabei zu beachten, dass für jede Konfliktressource $k \in \mathcal{R}^c(S)$ immer mindestens ein Paar $(k,i) \in \mathcal{R}^c(S) \times V_k$ existiert, das die Bedingung $r^u_{ik}(S_i) > \underline{r}^u_{ik}(W)$ erfüllt, da sich ansonsten ein Widerspruch zur Annahme $\underline{r}^c_k(W) \leq R_k$ ergeben würde. Im nächsten Schritt werden für die Zerlegung des Suchraums $\mathcal{S}_T(W)$ zwei direkte Nachfolgerknoten generiert. Dazu wird die Startzeitbeschränkung W_i partitioniert, d. h., W_i wird in zwei nichtleere disjunkte Mengen W'_i und W''_i zerlegt, sodass $W_i = W'_i \cup W''_i$ gilt. Die Partitionierung wird durch das Hinzufügen einer oberen sowie einer unteren Schranke für die Belegung der Ressource k durch Vorgang i erreicht. Im ersten Schritt wird dazu die obere Schranke $\bar{u}_{ik}$ für die Ressourcenbelegung von Vorgang i durch den abgerundeten Mittelwert der Ressourcenbelegung $r^u_{ik}(S_i)$ und der Mindestbelegung $\underline{r}^u_{ik}(W)$ berechnet. Anschließend werden die Startzeitbeschränkungen der direkten Nachfolgerknoten durch $W' := W$ und $W'' := W$ initialisiert und W'_i und W''_i den jeweiligen Teilmengen der Partitionierung durch $W'_i := \{\tau \in W_i \mid r^u_{ik}(\tau) > \bar{u}_{ik}\}$ und $W''_i := \{\tau \in W_i \mid r^u_{ik}(\tau) \leq \bar{u}_{ik}\}$ zugeordnet. Da Schedule S weiterhin dem Mini-

malpunkt des Suchraums $\mathcal{S}_T(W')$ durch $r^u_{ik}(S_i) > \bar{u}_{ik}$ entspricht, wird Knoten W' direkt der Menge Ω hinzugefügt. Im Gegensatz dazu, wird für Knoten W'' zunächst überprüft, ob der Suchraum einen W-zulässigen Schedule enthält, d. h., ob $\mathcal{S}_T(W'') \neq \emptyset$ erfüllt ist. Falls $\mathcal{S}_T(W'') \neq \emptyset$ gilt, wird Knoten W'' ebenfalls der Menge Ω hinzugefügt und in den folgenden Iterationen weiter untersucht. Es ist dabei zu beachten, dass der Suchraum von Knoten W durch den Verzweigungsschritt in zwei disjunkte Teilmengen $\mathcal{S}_T(W')$ und $\mathcal{S}_T(W'')$ zerlegt wird, sodass kein W-zulässiger Schedule im Enumerationsverlauf ausgeschlossen wird, d. h., es gilt $\mathcal{S}_T(W) = \mathcal{S}_T(W') \cup \mathcal{S}_T(W'')$. Als Folge daraus ergibt sich zum einen, dass sich der unzulässige Schedule S weiterhin im Suchraum genau eines direkten Nachfolgerknotens befindet, und zum anderen, dass jeder zulässige Schedule höchstens ein Mal im Enumerationsverlauf generiert wird. Das beschriebene Enumerationsverfahren wird solange fortgesetzt, bis alle Knoten im Enumerationsbaum betrachtet wurden ($\Omega = \emptyset$). Nach Ablauf des Enumerationsschemas gibt Algorithmus 5.3 schließlich die Menge aller im Enumerationsverlauf generierten Schedules Φ zurück.

Die Korrektheit von Algorithmus 5.3 folgt analog zum relaxationsbasierten Enumerationsschema aus der Vollständigkeit des Verfahrens bzw. aus Satz 5.3. Weiterhin wird durch Lemma 5.2 gezeigt, dass die Tiefe des Enumerationsbaums, der durch Algorithmus 5.3 generiert wird, polynomiell beschränkt ist, woraus schließlich auch die Endlichkeit des Enumerationsschemas folgt.

Lemma 5.2. *Der durch Algorithmus 5.3 generierte Enumerationsbaum hat eine maximale Tiefe von* $\mathcal{O}(|V||\mathcal{R}| \min(\overline{p}, |V||\mathcal{R}|\lceil \log_2 \overline{p} \rceil^2))$.

Beweis. Zunächst wird durch die Generierung eines Nachfolgerknotens in Algorithmus 5.3 entweder die minimal mögliche Belegung einer Ressource durch einen Vorgang erhöht (W') oder die maximal mögliche Belegung verringert (W''). Entsprechend können maximal $|V||\mathcal{R}|\overline{p}$ Verzweigungen im Enumerationsverlauf vorgenommen werden, bis schließlich ein Blatt des Suchbaums erreicht wird.

Im Folgenden wird ein beliebiger Weg $\mathcal{W} = (W^1, W^2, \ldots, W^s)$ im Enumerationsbaum mit W^μ als Startzeitbeschränkung des μ-ten Knotens auf dem Weg betrachtet. Zunächst sei angenommen, dass alle Enumerationsknoten $W^2, W^3, \ldots, W^s$ durch die Erhöhung einer minimal möglichen Ressourcenbelegung generiert werden (W'). Dann gilt $S = \min \mathcal{S}_T(W^1) = \min \mathcal{S}_T(W^2) = \ldots = \min \mathcal{S}_T(W^s) \notin \mathcal{S}_R$ für alle Knoten auf dem Weg $\mathcal{W}$. Da $\bar{u}_{ik}$ dem abgerundeten Mittelwert von $r^u_{ik}(S_i)$ und der minimal möglichen Ressourcenbelegung in jedem Verzweigungsschritt entspricht, folgt schließlich eine maximale Länge von $\mathcal{O}(|V||\mathcal{R}|\lceil \log_2 \overline{p} \rceil)$ für den Weg $\mathcal{W}$. Als Nächstes sei angenommen, dass $\mathcal{W}$ einen beliebigen Weg im Enumerationsbaum darstellt. Zunächst gilt, dass $\bar{u}_{ik}$

in jedem Verzweigungsschritt nicht größer ist als der Mittelwert der minimal möglichen und der maximal möglichen Belegung der betrachteten Ressource $k \in \mathcal{R}^c(S)$ durch Vorgang $i \in V_k$ über alle Startzeitpunkte in W_i. Entsprechend können auf dem Weg $\mathcal{W}$ höchstens $\mathcal{O}(|V||\mathcal{R}|\lceil\log_2 \overline{p}\rceil)$ Knoten liegen, für die eine maximal mögliche Ressourcenbelegung verringert wird (W''). Zusammenfassend folgt somit eine maximale Tiefe von $\mathcal{O}(|V|^2|\mathcal{R}|^2\lceil\log_2 \overline{p}\rceil^2)$ für den Enumerationsbaum. □

Satz 5.3. *Algorithmus 5.3 ist vollständig, d. h.,* $\mathcal{OS} \neq \emptyset \Rightarrow \Phi \cap \mathcal{OS} \neq \emptyset$.

Beweis. Zunächst gilt $\mathcal{S}_T(W) = \mathcal{S}_T \supseteq \mathcal{S}$ für den Suchraum des Wurzelknotens, sodass der gesamte Suchraum des Enumerationsverfahrens alle zulässigen Schedules enthält. Weiterhin ergibt sich aus dem Enumerationsverlauf in Algorithmus 5.3, dass über jeden Enumerationsknoten W mit $S = \min \mathcal{S}_T(W) \notin \mathcal{S}_R$ verzweigt wird, der mindestens einen zulässigen Schedule enthält. Dabei folgt aus $\mathcal{S}_T(W) = \mathcal{S}_T(W') \cup \mathcal{S}_T(W'')$, dass kein zulässiger Schedule durch eine Verzweigung ausgeschlossen wird. Da die Anzahl der Verzweigungen endlich ist (s. Lemma 5.2) und $S = \min \mathcal{S}_T(W)$ die Projektdauer auf der Menge $\mathcal{S}_T(W)$ minimiert, folgt aus $\mathcal{OS} \neq \emptyset \Rightarrow \mathcal{S} \neq \emptyset$ schließlich die Vollständigkeit von Algorithmus 5.3. □

Es ist zu beachten, dass die beschriebene Partitionierung der Startzeitbeschränkung eines Vorgangs in jedem Verzweigungsschritt des Enumerationsschemas in Algorithmus 5.3 lediglich die beste dem Autor dieser Arbeit bekannte Variante für die Partitionierung darstellt. Im Allgemeinen kann dagegen jede beliebige Partitionierung in einem Verzweigungsschritt zur Zerlegung eines Suchraums gewählt werden, wobei die Korrektheit des Enumerationsschemas stets gegeben ist.

5.2.2 Branch-and-Bound-Algorithmus

Im Folgenden wird das partitionsbasierte Enumerationsschema aus Abschnitt 5.2.1 zu einem Branch-and-Bound-Verfahren für das RCPSP/max-π erweitert. Da das partitionsbasierte Branch-and-Bound-Verfahren bis auf den Verzweigungsschritt identisch zum relaxationsbasierten Verfahren in Abschnitt 5.1.4 ist, beschränken sich die nachfolgenden Ausführungen weitestgehend auf die Verzweigung der Knoten im Enumerationsverlauf.

Zunächst wird die Suchstrategie des partitionsbasierten Branch-and-Bound-Verfahrens durch eine Traversierungs- und Verzweigungsstrategie eindeutig festgelegt. Im Gegensatz zum relaxationsbasierten Ansatz werden dabei keine unterschiedlichen Generierungs- oder Reihenfolgestrategien verwendet. Stattdessen werden in jedem Verzweigungsschritt

alle direkten Nachfolgerknoten auf einmal generiert und in der Reihenfolge, wie im Enumerationsschema in Algorithmus 5.3 beschrieben, dem Suchbaum hinzugefügt. Für die Auswahl des nächsten zu untersuchenden Knotens im Enumerationsbaum (Traversierungsstrategie) wird entweder eine Tiefensuche (DFS) oder eine Scattered-Path-Suche (SPS) verwendet. Für die Zerlegung des Suchraums eines Enumerationsknotens mit $S = \min \mathcal{S}_T(W) \notin \mathcal{S}_R$ wird durch die Verzweigungsstrategie eine Prioritätsregel festgelegt, nach der ein Paar $(k, i) \in \mathcal{R}^c(S) \times V_k$ ausgewählt wird. Im Folgenden werden Prioritätsregeln vorgestellt, die in experimentellen Voruntersuchungen die besten Ergebnisse für die Verzweigungsstrategie zeigen konnten. Die ersten Prioritätsregeln, die betrachtet werden, bestimmen die Konfliktressource k und den Vorgang $i \in V_k$ getrennt voneinander. Zunächst wird für die Auswahl der Konfliktressource die ACO- oder die RCO-Regel angewendet (s. Abschnitt 5.1.4). Nach der Auswahl der Konfliktressource k wird sowohl für die ACO- als auch die RCO-Regel jedem Vorgang $i \in V_k$ mit $r^u_{ik}(S_i) > \underline{r}^u_{ik}(W)$ ein Prioritätsregelwert $\pi_i = r^c_{ik}(S_i) - \underline{r}^c_{ik}(W)$ zugeordnet. Wie die Konfliktressource k wird auch der Vorgang i mit dem größten Prioritätsregelwert und dem kleinsten Index ausgewählt. Experimentelle Performance-Analysen konnten zeigen, dass die Prioritätsregeln ACO und RCO, die beide dazu tendieren, die Kapazitätsüberschreitungen über den Enumerationsverlauf schnell zu verringern, besonders gut für kleinere Probleminstanzen geeignet sind. Für größere Probleminstanzen konnten dagegen bessere Prioritätsregeln gefunden werden, die auf der Idee basieren, ein Paar $(k, i) \in \mathcal{R}^c(S) \times V_k$ auszuwählen, sodass die maximal verbleibende Kapazität $R_k - \underline{r}^c_k(W')$ des direkten Nachfolgerknotens W' tendenziell soweit wie möglich verringert wird. Es ist dabei zu beachten, dass diese Prioritätsregeln so interpretiert werden können, dass sie die Wahrscheinlichkeit erhöhen, dass ein direkter Nachfolgerknoten W' von den weiteren Untersuchungen im Enumerationsbaum durch $\underline{r}^c_k(W') > R_k$ ausgeschlossen werden kann, sodass eine geringere Breite des Suchbaums zu erwarten ist. Die vielversprechendste dieser Prioritätsregeln wird als RCR-Regel (engl. Remaining Capacity Reduction) bezeichnet und ordnet jedem Paar $(k, i) \in \mathcal{R}^c(S) \times V_k$ mit $r^u_{ik}(S_i) > \underline{r}^u_{ik}(W)$ einen Prioritätsregelwert

$$\pi_{ki} = \frac{(\bar{u}_{ik} + 1 - \underline{r}^u_{ik}(W))\, r^d_{ik}}{\max(R_k - \underline{r}^c_k(W); 0{,}1)}$$

zu. Dabei entspricht der Zähler der minimalen Steigerung von $\underline{r}^c_k(W)$ durch den Verzweigungsschritt in Bezug auf den direkten Nachfolgerknoten W' ($\underline{r}^c_k(W')$) und der Nenner der maximal verbleibenden Kapazität $R_k - \underline{r}^c_k(W)$ von Ressource k (oder 0,1 für den Fall $R_k - \underline{r}^c_k(W) = 0$). Für die RCR-Regel wird schließlich das Paar (k, i) mit dem größten Prioritätsregelwert für den Verzweigungsschritt gewählt, wobei im Fall gleicher Priori-

tätsregelwerte das Paar mit dem kleinsten Index der Ressource und des Vorgangs gewählt wird.

Algorithmus 5.4: Partitionsbasierter Branch-and-Bound-Algorithmus

Input: RCPSP/max-π-Instanz
Output: Optimaler Schedule S^*

Bestimme die Distanzmatrix $D = (d_{ij})_{i,j \in V}$
if $\mathcal{S}_T = \emptyset$ **then** terminate ($\mathcal{S} = \emptyset$)
Setze $ES_i := d_{0i}$, $LS_i := -d_{i0}$ und $W_i := [ES_i, LS_i] \cap \mathbb{Z}$ für alle $i \in V$
Führe Preprocessing-Schritt auf W aus
if $\mathcal{S}_T(W) = \emptyset$ **then** terminate ($\mathcal{S} = \emptyset$)
$S := \min \mathcal{S}_T(W), \quad LB := S_{n+1}$
$\Omega := \{(W, S, LB)\}, \quad UB := \bar{d} + 1$
while $\Omega \neq \emptyset$ **do**
  Entferne den obersten Knoten (W, S, LB) vom Stack Ω
  if $LB < UB$ **then**
    Führe Konsistenztests auf W aus
    if $\mathcal{S}_T^{UB}(W) \neq \emptyset$ **then**
      Aktualisiere $S := \min \mathcal{S}_T(W)$
      if $S \in \mathcal{S}_R$ **then**
        $S^* := S, \quad UB := S^*_{n+1}$
      else
        Wähle ein Paar $(k, i) \in \mathcal{R}^c(S) \times V_k$ mit $r^u_{ik}(S_i) > \underline{r}^u_{ik}(W)$ gemäß der Verzweigungsstrategie
        Berechne $\bar{u}_{ik} := \left\lfloor \frac{r^u_{ik}(S_i) + \underline{r}^u_{ik}(W)}{2} \right\rfloor$
        $W' := W, \qquad W'_i := \{\tau \in W_i \mid r^u_{ik}(\tau) > \bar{u}_{ik}\}$
        Setze $S' := S$, $LB' := LB$ und lege (W', S', LB') auf den Stack Ω
        $W'' := W, \qquad W''_i := \{\tau \in W_i \mid r^u_{ik}(\tau) \leq \bar{u}_{ik}\}$
        if $\mathcal{S}_T^{UB}(W'') \neq \emptyset$ **then**
          $S'' := \min \mathcal{S}_T(W'')$
          if $S'' \in \mathcal{S}_R$ **then**
            $S^* := S'', \quad UB := S^*_{n+1}$
          else
            Setze $LB'' := S''_{n+1}$ und lege (W'', S'', LB'') auf den Stack Ω
if $UB = \bar{d} + 1$ **then** terminate ($\mathcal{S} = \emptyset$)
else return S^*

Das partitionsbasierte Branch-and-Bound-Verfahren wird in Algorithmus 5.4 beschrieben. Wie bereits für das relaxationsbasierte Verfahren in Abschnitt 5.1.4 beschränkt sich die Darstellung des Algorithmus auf eine Tiefensuche. Der partitionsbasierte Branch-and-Bound-Algorithmus unterscheidet sich vom relaxationsbasierten Verfahren in Algorithmus 5.2, abgesehen vom Verzweigungsschritt, lediglich dadurch, dass angenommen wird,

dass in Zeile 11 entweder der Fixpunkt der Konsistenztests aus der Menge Γ^{T} oder Γ^{B} bestimmt wird (s. Abschnitt 4.2). Entsprechend wird im Folgenden nur noch auf die Zerlegung des Suchraums eines Enumerationsknotens eingegangen. Durch die Annahme, dass der Fixpunkt der Konsistenztests aus der Menge Γ^{T} oder Γ^{B} für jeden Knoten bestimmt wird, folgt zunächst, dass die Bedingung $\nexists k \in \mathcal{R} : \underline{r}_k^c(W) > R_k$ (s. Algorithmus 5.3) in jedem Verzweigungsschritt erfüllt ist. Aus diesem Grund ist sichergestellt, dass immer mindestens ein Paar $(k,i) \in \mathcal{R}^c(S) \times V_k$ für eine Konfliktressource k mit $r_{ik}^u(S_i) > \underline{r}_{ik}^u(W)$ existiert, die gemäß der Verzweigungsstrategie ausgewählt wird. Die Zerlegung des Suchraums eines Enumerationsknotens erfolgt dabei analog zu Algorithmus 5.3. Während der direkte Nachfolgerknoten W' immer auf den Stack Ω gelegt wird, kann Knoten W'' von den weiteren Untersuchungen ausgeschlossen werden, falls $\mathcal{S}_T^{UB}(W'') = \emptyset$ gilt. Für den Fall, dass dagegen $\mathcal{S}_T^{UB}(W'') \neq \emptyset$ gegeben ist, wird der Minimalpunkt S'' von $\mathcal{S}_T(W'')$ berechnet und die Ressourcenzulässigkeit von S'' überprüft. Gilt $S'' \in \mathcal{S}_R$, wird $S^* := S''$ gesetzt und UB entsprechend aktualisiert, wohingegen andernfalls der direkte Nachfolgerknoten (W'', S'', LB'') mit $LB'' := S''_{n+1}$ auf den Stack Ω gelegt wird, um in weiteren Iterationsschritten untersucht zu werden. Nach der Untersuchung aller Enumerationsknoten ($\Omega = \emptyset$) gibt Algorithmus 5.4 schließlich einen optimalen Schedule S^* zurück oder zeigt, falls $UB = \bar{d}+1$ gilt, die Unlösbarkeit der betrachteten Probleminstanz ($\mathcal{S} = \emptyset$).

5.3 Ein konstruktionsbasierter Ansatz

In Abschnitt 5.3.1 wird ein konstruktionsbasiertes Enumerationsschema für das RCPSP/max-π vorgestellt, das durch eine sukzessive Ein- und Ausplanung von Vorgängen in jedem Enumerationsschritt eine Menge zulässiger Schedules generiert. Zur Bestimmung der Einplanungszeitpunkte der Vorgänge in jedem Enumerationsschritt werden in Abschnitt 5.3.2 verschiedene Verfahren beschrieben. In Abschnitt 5.3.3 wird gezeigt, wie Redundanzen im Enumerationsverlauf vermieden werden können, und in Abschnitt 5.3.4 wird schließlich das konstruktionsbasierte Branch-and-Bound-Verfahren aus dem Enumerationsansatz entwickelt.

5.3.1 Enumerationsschema

Im Folgenden wird das Enumerationsschema für das konstruktionsbasierte Branch-and-Bound-Verfahren, das in Abschnitt 5.3.4 beschrieben wird, vorgestellt. Das Enumerationsverfahren basiert auf einer sukzessiven Einplanung aller Vorgänge des Projekts, wobei

durch einen Ausplanungsschritt die Einhaltung von Zeitabständen zwischen den Vorgängen sichergestellt wird. Das Konzept der Ausplanung von Vorgängen stammt aus der Arbeit Franck et al. (2001b), in der das serielle Generierungsschema für das RCPSP aus Kolisch (1996) um einen Ausplanungsschritt erweitert wird, um zeitliche Höchstabstände zwischen den Vorgängen für das RCPSP/max zu berücksichtigen.

Im weiteren Verlauf wird zunächst das konstruktionsbasierte Enumerationsschema ohne die Berücksichtigung von Startzeitbeschränkungen entwickelt. Im Anschluss daran wird das konstruktionsbasierte Enumerationsschema soweit angepasst, dass auch Startzeitbeschränkungen eingebunden werden, sodass in jedem Enumerationsknoten die beschriebenen Verfahren aus Kapitel 4 angewendet werden können.

Das konstruktionsbasierte Enumerationsschema ohne die Einbindung von Startzeitbeschränkungen wird in Algorithmus 5.5 beschrieben. In dem Verfahren wird jeder Enumerationsknoten durch ein Paar $(\mathcal{C}, S)$ dargestellt. Die Menge $\mathcal{C} \subseteq V$ entspricht dabei der Menge aller aktuell eingeplanten Vorgänge und S einem zeitzulässigen Schedule, der die Einplanungszeitpunkte aller Vorgänge $i \in \mathcal{C}$ und die frühestmöglichen zeitzulässigen Startzeitpunkte aller aktuell nicht eingeplanten Vorgänge $i \in V \setminus \mathcal{C}$ unter Berücksichtigung der Startzeitpunkte aller aktuell eingeplanten Vorgänge angibt. Für die nachfolgenden Erläuterungen wird zunächst in Anlehnung an Definition 2.6.3 in Neumann et al. (2003) ein sogenannter Teilschedule eingeführt, um die Startzeitpunkte der Vorgänge zu beschreiben, die in einem Iterationsschritt von Algorithmus 5.5 eingeplant sind.

Definition 5.1. *$S^{\mathcal{C}} := (S_i)_{i \in \mathcal{C}}$ mit $\mathcal{C} \subseteq V$, $0 \in \mathcal{C}$ und $S_i \in \mathbb{Z}_{\geq 0}$ für alle $i \in \mathcal{C}$ wird Teilschedule genannt. Falls $S_j \geq S_i + \delta_{ij}$ für alle $(i,j) \in E \cap \mathcal{C} \times \mathcal{C}$ und $S_0 = 0$ gilt, wird Teilschedule $S^{\mathcal{C}}$ als zeitzulässig bezeichnet und im Fall $\sum_{i \in \mathcal{C}} r^c_{ik}(S_i) \leq R_k$ für alle $k \in \mathcal{R}$ ressourcenzulässig genannt. Ein zeit- und ressourcenzulässiger Teilschedule $S^{\mathcal{C}}$ heißt zulässig. $S^{\mathcal{C} \cup \{i\}}$ mit $i \in V \setminus \mathcal{C}$ wird Erweiterung von $S^{\mathcal{C}}$ um Vorgang i genannt, wobei S_i als zeitzulässig, ressourcenzulässig oder zulässig bezeichnet wird, falls $S^{\mathcal{C} \cup \{i\}}$ zeitzulässig, ressourcenzulässig oder zulässig ist.*

Im Initialisierungsschritt des konstruktionsbasierten Enumerationsschemas wird zunächst die Distanzmatrix D bestimmt, aus der die frühesten und spätesten zeitzulässigen Schedules ES und LS abgeleitet werden. Anschließend wird der Wurzelknoten $(\mathcal{C}, S)$ mit $\mathcal{C} := \{0\}$ und $S := ES$ der Menge Ω hinzugefügt und die Menge Φ durch $\Phi := \emptyset$ initialisiert. In jeder Iteration wird ein Knoten $(\mathcal{C}, S)$ aus der Menge Ω entnommen. Falls $\mathcal{C} \neq V$ gilt, wird ein Vorgang $i \in \bar{\mathcal{C}}$ aus der Menge aller aktuell nicht eingeplanten Vorgänge $\bar{\mathcal{C}} := V \setminus \mathcal{C}$ ausgewählt. Für den ausgewählten Vorgang werden alle ressourcenzulässigen Startzeitpunkte in der Menge $\{S_i, S_i + 1, \ldots, -d_{i0}\}$ bestimmt und in der sogenannten

Algorithmus 5.5: Konstruktionsbasiertes Enumerationsschema

Input: RCPSP/max-π-Instanz
Output: Menge Φ aller generierten Schedules

Bestimme die Distanzmatrix $D = (d_{ij})_{i,j \in V}$
if $\mathcal{S}_T = \emptyset$ **then** terminate ($\mathcal{S} = \emptyset$)
$ES_i := d_{0i}$, $LS_i := -d_{i0}$ für alle $i \in V$
$\mathcal{C} := \{0\}, \quad S := ES$
$\Omega := \{(\mathcal{C}, S)\}, \quad \Phi := \emptyset$
while $\Omega \neq \emptyset$ **do**
  Entferne $(\mathcal{C}, S)$ aus der Menge Ω
  if $\mathcal{C} = V$ **then**
    $\Phi := \Phi \cup \{S\}$
  else
    Wähle einen Vorgang $i \in \bar{\mathcal{C}}$
    $\Theta_i := \{\tau \in \{S_i, S_i + 1, \ldots, -d_{i0}\} \mid r_k^c(S^{\mathcal{C}}) + r_{ik}^c(\tau) \leq R_k$ für alle $k \in \mathcal{R}_i\}$
    Berechne T_i (Algorithmus 5.7 oder 5.8)
    forall $t \in T_i$ **do**
      $S_i' := t, \quad S' := (\max(S_j, S_i' + d_{ij}))_{j \in V}$
      if $\exists j \in \mathcal{C} : S_j' > S_j$ **then**
        $\mathcal{C}' := \mathcal{C} \setminus \{j \in \mathcal{C} \mid S_j' > S_j\}$
      else
        $\mathcal{C}' := \mathcal{C} \cup \{i\}$
      $\Omega := \Omega \cup \{(\mathcal{C}', S')\}$
return Φ

Menge aller Einplanungszeitpunkte Θ_i von Vorgang i abgespeichert. Die Ressourcenzulässigkeit aller Zeitpunkte in Θ_i wird dabei durch die Berücksichtigung der Ressourceninanspruchnahme $r_k^c(S^{\mathcal{C}}) := \sum_{j \in \mathcal{C}} r_{jk}^c(S_j)$ jeder Ressource $k \in \mathcal{R}_i$ durch den gegebenen Teilschedule $S^{\mathcal{C}}$ sichergestellt. In Schirmer (1999, Theorem 9.5) wurde für das RCPSP/π bereits gezeigt, dass die sukzessive Einplanung von Vorgängen zu ihren jeweils frühestmöglichen zulässigen Startzeitpunkten im Allgemeinen nicht dazu führt, dass für jede Probleminstanz ein optimaler Schedule generiert werden kann. Basierend darauf wurden in der Literatur zum RCPSP/π ausschließlich serielle Generierungsschemata entwickelt, die in jedem Iterationsschritt den Startzeitpunkt eines noch nicht eingeplanten Vorgangs aus der Menge aller in der Planungssituation zulässigen Startzeitpunkte auswählen, wobei teilweise auch Konsistenztests verwendet werden (s. Böttcher et al., 1999; Schirmer, 1999; Alvarez-Valdes et al., 2006, 2008, 2015). Es ist zu beachten, dass die Enumeration über alle Startzeitpunkte in Θ_i in jedem Verzweigungsschritt von Algorithmus 5.5 ohne die Berücksichtigung von Ausplanungsschritten bereits ein vollständiges Verfahren für das RCPSP/max-π darstellen würde. Dabei könnte zudem $LS_i = -d_{i0}$ bei der Bestim-

mung von Θ_i durch einen planungsabhängigen spätesten zeitzulässigen Startzeitpunkt $LS_i(S^{\mathcal{C}}) := \min(-d_{i0}, \min_{j \in \mathcal{C}}(S_j - d_{ij}))$ in jedem Verzweigungsschritt ersetzt werden. Da diese Art der Enumeration allerdings viele Schedules generieren würde, die aufgrund der betrachteten Zielsetzung der Projektdauerminimierung bereits im Vorhinein ausgeschlossen werden können, wird im Folgenden eine alternative Variante beschrieben. Zunächst wird dafür gezeigt, dass es ausreichend ist, eine Teilmenge von Θ_i für die Einplanung eines Vorgangs $i \in \bar{\mathcal{C}}$ in jedem Iterationsschritt zu betrachten, sodass weiterhin die Generierung mindestens eines optimalen Schedules für eine lösbare Probleminstanz garantiert ist. Es ist dabei zu beachten, dass durch die Einschränkung von Θ_i aufgrund der zeitlichen Höchstabstände zwischen den Vorgängen des Projekts zusätzlich die Ausplanung von Vorgängen im Enumerationsverlauf notwendig wird. Die entsprechende Teilmenge

$$T_i := \{\tau \in \Theta_i \mid \nexists \tau' \in [0, \tau[\cap \Theta_i : r^u_{ik}(\tau) \geq r^u_{ik}(\tau') \text{ für alle } k \in \mathcal{R}_i\}$$

von Θ_i wird als eingeschränkte Menge aller Einplanungszeitpunkte von Vorgang i oder auch als Einschränkung von Θ_i bezeichnet. Durch die Verwendung der Einschränkung T_i wird entsprechend nur dann ein späterer als der frühestmögliche zulässige Startzeitpunkt eines Vorgangs $i \in \bar{\mathcal{C}}$ als möglicher Einplanungszeitpunkt betrachtet, falls dadurch im Vergleich zu jedem früheren Zeitpunkt in Θ_i die Inanspruchnahme mindestens einer Ressource $k \in \mathcal{R}_i$ verringert werden kann. Ein späterer als der frühestmögliche zulässige Einplanungszeitpunkt in T_i kann somit als Möglichkeit interpretiert werden, um Kapazitäten von Ressourcen für die Einplanung weiterer Vorgänge freizuhalten. Zunächst sei angenommen, dass die Einschränkung T_i gegeben ist, wobei nachfolgend in Abschnitt 5.3.2 zwei verschiedene Algorithmen zur Berechnung von T_i vorgestellt werden. Basierend auf der Einschränkung T_i des ausgewählten Vorgangs $i \in \bar{\mathcal{C}}$ werden schließlich alle direkten Nachfolger für den Enumerationsknoten $(\mathcal{C}, S)$ generiert. Für jeden direkten Nachfolgerknoten $(\mathcal{C}', S')$ wird Vorgang i zunächst einem Zeitpunkt $t \in T_i$ durch $S'_i := t$ zugeordnet. Anschließend werden die frühesten zeitzulässigen Startzeitpunkte für alle Vorgänge des Projekts, unter Berücksichtigung, dass Vorgang i zum Zeitpunkt t startet, durch $S' := (\max(S_j, S'_i + d_{ij}))_{j \in V}$ aktualisiert. Da entsprechend jeder aktuell eingeplante Vorgang den Zeitabstand zu jedem Zeitpunkt $t \in T_i$ einhält, d. h., $t \geq S_j + d_{ji}$ für alle $j \in \mathcal{C}$ gilt, existiert für den Fall, dass Zeitpunkt t nicht zeitzulässig ist, mindestens ein Vorgang $j \in \mathcal{C}$ mit $t > S_j - d_{ij}$. Entsprechend wird in diesem Fall die zulässige Einplanung des Vorgangs i durch den induzierten spätesten Startzeitpunkt $LS_i(S_j) := S_j - d_{ij}$ mindestens eines Vorgangs $j \in \mathcal{C}$ verhindert. Um schließlich im weiteren Enumerationsverlauf die Zeitzulässigkeit der Erweiterung $S'^{\mathcal{C} \cup \{i\}}$ mit $S'_i = t$ zu erreichen, werden alle Vorgänge $j \in \mathcal{C}$ mit $t > LS_i(S_j)$ bzw. $S'_j > S_j$ durch $\mathcal{C}' := \mathcal{C} \setminus \{j \in \mathcal{C} \mid S'_j > S_j\}$ aus-

geplant. Es ist dabei zu beachten, dass $0 \in \mathcal{C}'$ durch $t \leq LS_i$ aufgrund der Definition von Θ_i in jedem Iterationsschritt sichergestellt ist. Falls Zeitpunkt t zeitzulässig ist und somit $t \leq \min_{j \in \mathcal{C}} LS_i(S_j)$ erfüllt ist, wird Vorgang i zum Zeitpunkt t durch $\mathcal{C}' := \mathcal{C} \cup \{i\}$ eingeplant. Nach dem Einplanungs- oder Ausplanungsschritt wird schließlich der direkte Nachfolgerknoten $(\mathcal{C}', S')$ der Menge Ω hinzugefügt, um im weiteren Enumerationsverlauf untersucht zu werden. Aus dem beschriebenen Verfahrensablauf geht hervor, dass jeder Teilschedule $S^{\mathcal{C}}$ eines Enumerationsknotens $(\mathcal{C}, S)$ zulässig ist. Entsprechend wird für jeden Enumerationsknoten $(\mathcal{C}, S)$ mit $\mathcal{C} = V$, der aus der Menge Ω entnommen wird, der Schedule $S = S^{\mathcal{C}} \in \mathcal{S}$ in der Menge Φ abgespeichert. Nachdem alle Enumerationsknoten untersucht wurden ($\Omega = \emptyset$), endet der Algorithmus 5.5 und gibt schließlich die Menge aller generierten Schedules Φ zurück.

Im Folgenden wird gezeigt, dass Algorithmus 5.5 nach endlich vielen Iterationen beendet wird und mindestens einen optimalen Schedule generiert, wenn mindestens eine optimale Lösung existiert. Es ist dabei zu beachten, dass daraus die Korrektheit von Algorithmus 5.5 direkt abgeleitet werden kann, da jeder generierte Schedule zulässig ist ($\Phi \subseteq \mathcal{S}$) und $\mathcal{OS} \neq \emptyset \Leftrightarrow \mathcal{S} \neq \emptyset$ aus der Endlichkeit des zulässigen Bereichs folgt. Zunächst generiert Algorithmus 5.5 nach Satz 5.4 die Menge aller aktiven Schedules $\mathcal{AS}$ einer Probleminstanz, wobei ein Schedule S genau dann als aktiv bezeichnet wird, wenn er zulässig ist und kein anderer zulässiger Schedule S' mit $S' \leq S$ existiert (vgl. Neumann et al., 2000). Da für jede lösbare Probleminstanz ($\mathcal{OS} \neq \emptyset$) mindestens ein optimaler Schedule existiert, der aktiv ist, folgt aus Satz 5.4 die Vollständigkeit von Algorithmus 5.5 bzw. $\mathcal{OS} \neq \emptyset \Rightarrow \Phi \cap \mathcal{OS} \neq \emptyset$.

Lemma 5.3. *Sei T_i die Einschränkung der Menge aller Einplanungszeitpunkte Θ_i von Vorgang $i \in V$. Dann gilt*

$$T_i = T_i^{\cup} := \bigcup_{\tau \in \Theta_i} \{\min\{\tau' \in \Theta_i \mid r_{ik}^u(\tau') \leq r_{ik}^u(\tau) \text{ für alle } k \in \mathcal{R}_i\}\}.$$

Beweis. Betrachtet sei ein Zeitpunkt $t \in T_i$. Dann kann $t = \min\{\tau \in \Theta_i \mid r_{ik}^u(\tau) \leq r_{ik}^u(t)$ für alle $k \in \mathcal{R}_i\}$ direkt aus der Definition von T_i abgeleitet werden, sodass $T_i \subseteq T_i^{\cup}$ gilt. Als Nächstes seien ein Zeitpunkt $\tau \in \Theta_i$ und ein Zeitpunkt $t := \min\{\tau' \in \Theta_i \mid r_{ik}^u(\tau') \leq r_{ik}^u(\tau)$ für alle $k \in \mathcal{R}_i\}$ gegeben, wobei $t \notin T_i$ angenommen wird. Da sich $t > \min\{\tau' \in \Theta_i \mid r_{ik}^u(\tau') \leq r_{ik}^u(t)$ für alle $k \in \mathcal{R}_i\}$ aus $t \notin T_i$ ergibt, was im Widerspruch zur Definition von t steht, folgt schließlich $T_i \supseteq T_i^{\cup}$ bzw. $T_i = T_i^{\cup}$. □

Lemma 5.4. *Sei $S^f \in \mathcal{S}$ ein beliebiger zulässiger Schedule und $(\mathcal{C}, S)$ ein Knoten im Enumerationsverlauf von Algorithmus 5.5 mit $\mathcal{C} \neq V$, $S \leq S^f$ und $r_{jk}^u(S_j) \leq r_{jk}^u(S_j^f)$ für alle $j \in \mathcal{C}$ und alle $k \in \mathcal{R}_j$. Dann existiert mindestens ein direkter Nachfolgerknoten*

$(\mathcal{C}', S')$, *der die Bedingungen* $S' \leq S^f$ *und* $r^u_{jk}(S'_j) \leq r^u_{jk}(S^f_j)$ *für alle* $j \in \mathcal{C}'$ *und alle* $k \in \mathcal{R}_j$ *erfüllt.*

Beweis. Gegeben sei ein Vorgang $i \in \bar{\mathcal{C}}$, der für die Generierung aller direkten Nachfolgerknoten von $(\mathcal{C}, S)$ ausgewählt wurde. Dann kann zunächst $S^f_i \in \Theta_i$ aus den Bedingungen $S_i \leq S^f_i \leq LS_i$ und $r^c_k(S^{\mathcal{C}}) + r^c_{ik}(S^f_i) \leq r^c_k(S^f) \leq R_k$ für alle $k \in \mathcal{R}_i$ abgeleitet werden. Aus Lemma 5.3 folgt schließlich, dass ein Einplanungszeitpunkt $t := \min\{\tau \in \Theta_i | r^u_{ik}(\tau) \leq r^u_{ik}(S^f_i)$ für alle $k \in \mathcal{R}_i\} \in T_i$ für den Vorgang i existiert $(S^f_i \in \Theta_i)$, sodass die Bedingungen $t \leq S^f_i$ und $r^u_{ik}(t) \leq r^u_{ik}(S^f_i)$ für alle $k \in \mathcal{R}_i$ erfüllt sind. Sei $(\mathcal{C}', S')$ nun der direkte Nachfolgerknoten von $(\mathcal{C}, S)$, der dem Zeitpunkt t zugeordnet ist. Dann folgt zunächst $S' \leq S^f$ aus $t \leq S^f_i$ $(t + d_{ij} \leq S^f_i + d_{ij} \leq S^f_j$ für alle $j \in V)$. Weiterhin werden die Bedingungen $r^u_{jk}(S'_j) \leq r^u_{jk}(S^f_j)$ für alle $j \in \mathcal{C}'$ und alle $k \in \mathcal{R}_j$ sowohl bei der Einplanung von Vorgang i $(\mathcal{C}' := \mathcal{C} \cup \{i\})$ als auch bei der Ausplanung von Vorgängen erfüllt. □

Satz 5.4. *Algorithmus 5.5 generiert alle aktiven Schedules, d. h.,* $\Phi \supseteq \mathcal{AS}$.

Beweis. Gegeben sei ein beliebiger aktiver Schedule $S^a \in \mathcal{AS}$. Dann gelten für den Wurzelknoten $(\mathcal{C}, S)$ zunächst die Bedingungen $S \leq S^a$ und $r^u_{jk}(S_j) \leq r^u_{jk}(S^a_j)$ für alle $j \in \mathcal{C}$ und alle $k \in \mathcal{R}_j$. Entsprechend folgt aus Lemma 5.4, dass mindestens ein vom Wurzelknoten ausgehender Weg im Enumerationsbaum existiert, auf dem jeder Knoten $(\mathcal{C}, S)$ die Bedingungen $S \leq S^a$ und $r^u_{jk}(S_j) \leq r^u_{jk}(S^a_j)$ für alle $j \in \mathcal{C}$ und alle $k \in \mathcal{R}_j$ erfüllt. Da bei der Generierung eines Nachfolgerknotens entweder der Startzeitpunkt S_j für mindestens einen Vorgang $j \in \mathcal{C}$ erhöht $(S'_j > S_j)$ oder Vorgang $i \in \bar{\mathcal{C}}$ eingeplant wird $(\mathcal{C}' := \mathcal{C} \cup \{i\})$, hat jeder Weg im Enumerationsbaum eine endliche Länge. Da zudem die Generierung eines Schedules $S \leq S^a$ mit $S \neq S^a$ der Annahme $S^a \in \mathcal{AS}$ widersprechen würde, folgt schließlich $\Phi \supseteq \mathcal{AS}$. □

Abschließend kann aus Lemma 5.5 abgeleitet werden, dass das Enumerationsschema in Algorithmus 5.5 nach endlich vielen Iterationen beendet wird.

Lemma 5.5. *Algorithmus 5.5 generiert maximal* $(\bar{d}+1)^{|V|(\bar{d}+1)}$ *Enumerationsknoten.*

Beweis. Für die Generierung eines Nachfolgerknotens im Enumerationsschema von Algorithmus 5.5 wird entweder der ausgewählte Vorgang $i \in \bar{\mathcal{C}}$ eingeplant oder der frühestmögliche Startzeitpunkt eines Vorgangs um mindestens eine Zeiteinheit erhöht. Da die Anzahl der möglichen Startzeitpunkte für jeden Vorgang durch $\bar{d}+1$ beschränkt ist, folgt eine maximale Tiefe von $|V|(\bar{d}+1)$ für den Enumerationsbaum. Entsprechend werden maximal $(\bar{d}+1)^{|V|(\bar{d}+1)}$ Enumerationsknoten in Algorithmus 5.5 generiert, da die Anzahl der Startzeitpunkte in T_i bzw. die Anzahl der direkten Nachfolgerknoten in jedem Verzweigungsschritt ebenfalls durch $\bar{d}+1$ beschränkt ist. □

Als Nächstes wird das Enumerationsschema in Algorithmus 5.5 um Startzeitbeschränkungen ergänzt, wodurch die Konsistenztests und unteren Schranken aus Kapitel 4 sowie Verfahren zur Redundanzvermeidung, die in Abschnitt 5.3.3 behandelt werden, im Enumerationsverlauf verwendet werden können. Das angepasste Enumerationsschema, das in Algorithmus 5.6 abgebildet ist, dient schließlich als Basis des konstruktionsbasierten Branch-and-Bound-Verfahrens, das in Abschnitt 5.3.4 entwickelt wird.

Definition 5.2. *Ein Teilschedule $S^{\mathcal{C}} = (S_i)_{i\in\mathcal{C}}$ heißt W-zulässig, wenn mindestens ein Schedule $S' \in \mathcal{S}_T(W)$ existiert, der die Bedingung $S'_i = S_i$ für alle Vorgänge $i \in \mathcal{C}$ erfüllt. Weiterhin wird der Zeitpunkt S_i eines Vorgangs $i \in \bar{\mathcal{C}}$ als W-zulässig bezeichnet, falls $S^{\mathcal{C}\cup\{i\}}$ W-zulässig ist.*

Algorithmus 5.6: Erweitertes konstruktionsbasiertes Enumerationsschema

Input: RCPSP/max-π-Instanz
Output: Menge Φ aller generierten Schedules

Bestimme die Distanzmatrix $D = (d_{ij})_{i,j\in V}$
if $\mathcal{S}_T = \emptyset$ **then** terminate ($\mathcal{S} = \emptyset$)
$ES_i := d_{0i}$, $LS_i := -d_{i0}$ und $W_i := [ES_i, LS_i] \cap \mathbb{Z}$ für alle $i \in V$
$\mathcal{C} := \{0\}, \quad S := ES$
$\Omega := \{(\mathcal{C}, S, W)\}, \quad \Phi := \emptyset$
while $\Omega \neq \emptyset$ **do**
  Entferne $(\mathcal{C}, S, W)$ aus der Menge Ω
  if $\mathcal{C} = V$ **then**
    $\Phi := \Phi \cup \{S\}$
  else
    Wähle einen Vorgang $i \in \bar{\mathcal{C}}$
    $\Theta_i := \{\tau \in W_i \mid r^c_k(S^{\mathcal{C}}) + r^c_{ik}(\tau) \leq R_k \text{ für alle } k \in \mathcal{R}_i\}$
    Berechne T_i (Algorithmus 5.7 oder 5.8)
    forall $t \in T_i$ **do**
      $S'_i := t, \quad S' := \min \widetilde{\mathcal{S}}_T(W, i, S'_i), \quad W' := (W_j \setminus [0, S'_j[)_{j\in V}$
      if $\exists j \in \mathcal{C} : S'_j > S_j$ **then**
        $\mathcal{C}' := \mathcal{C} \setminus \{j \in \mathcal{C} \mid S'_j > S_j\}$
      else if $S'_i = t$ **then**
        $\mathcal{C}' := \mathcal{C} \cup \{i\}$
      $\Omega := \Omega \cup \{(\mathcal{C}', S', W')\}$
return Φ

Für die Erweiterung des konstruktionsbasierten Enumerationsschemas in Algorithmus 5.6 wird zusätzlich eine Startzeitbeschränkung W für jeden Enumerationsknoten abgespeichert, sodass jeder Knoten durch ein Tripel $(\mathcal{C}, S, W)$ beschrieben wird. Wie für das

relaxations- und das partitionsbasierte Enumerationsschema wird die Startzeitbeschränkung W im Wurzelknoten durch $W_i := [ES_i, LS_i] \cap \mathbb{Z}$ für alle Vorgänge $i \in V$ initialisiert, sodass $\mathcal{S}_T(W) \supseteq \mathcal{S}$ gilt. Weiterhin werden die Menge aller Einplanungszeitpunkte Θ_i sowie die zugehörige Einschränkung T_i in jeder Iteration für den ausgewählten Vorgang $i \in \bar{\mathcal{C}}$ auf die Zeitpunkte in W_i beschränkt. Wie auch in Algorithmus 5.5 wird für die Generierung eines direkten Nachfolgerknotens $(\mathcal{C}', S', W')$ ein Zeitpunkt t aus der Einschränkung T_i entnommen und durch $S'_i := t$ Vorgang i zugeordnet. Im Anschluss daran wird der frühestmögliche W-zulässige Schedule S' bestimmt, der die Bedingungen $S' \geq S$ und $S'_i \geq t$ erfüllt. Da $S = (\min W_i)_{i \in V}$ für jeden Enumerationsknoten gilt, wird der frühestmögliche W-zulässige Schedule, der die Bedingungen $S' \geq S$ und $S'_i \geq t$ erfüllt, durch $S' := \min \widetilde{\mathcal{S}}_T(W, i, t)$ berechnet. Falls mindestens ein Vorgang $j \in \mathcal{C}$ mit $ES_j(W, i, t) > S_j$ $(S'_j > S_j)$ vorliegt, woraus $t > LS_i(W, j, S_j)$ folgt, ist der Startzeitpunkt t von Vorgang i nicht W-zulässig. Entsprechend werden in diesem Fall alle Vorgänge $j \in \mathcal{C}$ mit $S'_j > S_j$ ausgeplant, sodass Vorgang i im weiteren Enumerationsverlauf W-zulässig zum Zeitpunkt t eingeplant werden kann, soweit $t = ES_i(W, i, t)$ $(S'_i = t)$ gilt. Es ist zu beachten, dass aufgrund der Betrachtung von Startzeitbeschränkungen Vorgang i nur dann W-zulässig zum Zeitpunkt t eingeplant werden kann, wenn weder Vorgänge ausgeplant werden noch $S'_i > t$ gilt. Entsprechend wird Vorgang i nur dann zum Zeitpunkt t eingeplant $(\mathcal{C}' := \mathcal{C} \cup \{i\})$, falls die Bedingung $S'_i = t$ erfüllt ist.

Abschließend sei erwähnt, dass die Algorithmen 5.5 und 5.6 im Hinblick auf die Suchräume der Knoten identische Enumerationsbäume generieren. Dieser Zusammenhang ergibt sich direkt aus Satz 4.3, aus dem $\min \widetilde{\mathcal{S}}_T(W, i, t) = (\max(S_j, t + d_{ij}))_{j \in V}$ mit $S = (\min W_i)_{i \in V}$ für jeden Enumerationsknoten in Algorithmus 5.6 abgeleitet werden kann.

Die Korrektheit von Algorithmus 5.6 kann analog zu Algorithmus 5.5 gezeigt werden. Zunächst gibt Satz 5.5 an, dass Algorithmus 5.6 alle aktiven Schedules einer Probleminstanz generiert, wobei der Satz auf Lemma 5.6, einer Verallgemeinerung von Lemma 5.4, aufbaut. Abschließend folgt aus Satz 5.5 und daraus, dass jeder generierte Schedule $S \in \Phi$ zulässig ist und $\mathcal{OS} \neq \emptyset \Leftrightarrow \mathcal{S} \neq \emptyset$ gilt, die Korrektheit von Algorithmus 5.6. Weiterhin kann die Endlichkeit des Verfahrens analog zu Lemma 5.5 gezeigt werden.

Lemma 5.6. *Sei $S^f \in \mathcal{S}$ ein beliebiger zulässiger Schedule und $(\mathcal{C}, S, W)$ ein Knoten im Enumerationsverlauf von Algorithmus 5.6 mit $\mathcal{C} \neq V$, $S \leq S^f$, $S^f \in \mathcal{S}_T(W)$ und $r^u_{jk}(S_j) \leq r^u_{jk}(S^f_j)$ für alle $j \in \mathcal{C}$ und alle $k \in \mathcal{R}_j$. Dann existiert mindestens ein direkter Nachfolgerknoten $(\mathcal{C}', S', W')$, der die Bedingungen $S' \leq S^f$, $S^f \in \mathcal{S}_T(W')$ und $r^u_{jk}(S'_j) \leq r^u_{jk}(S^f_j)$ für alle $j \in \mathcal{C}'$ und alle $k \in \mathcal{R}_j$ erfüllt.*

Beweis. Analog zum Beweis von Lemma 5.4 kann aus Lemma 5.3 abgeleitet werden, dass für einen Vorgang $i \in \bar{\mathcal{C}}$, der für die Verzweigung von Knoten $(\mathcal{C}, S, W)$ ausgewählt wird, immer ein Zeitpunkt $t \in T_i$ existiert, der die Bedingungen $t \leq S_i^f$ und $r_{ik}^u(t) \leq r_{ik}^u(S_i^f)$ für alle $k \in \mathcal{R}_i$ erfüllt. Entsprechend gilt für den direkten Nachfolgerknoten $(\mathcal{C}', S', W')$, der dem Zeitpunkt t zugeordnet ist, $S' := \min \widetilde{\mathcal{S}}_T(W, i, t) \leq S^f$ und somit $S^f \in \mathcal{S}_T(W')$ mit $W' := (W_j \setminus [0, S'_j[)_{j \in V}$. Weiterhin werden die Bedingungen $r_{jk}^u(S'_j) \leq r_{jk}^u(S_j^f)$ für alle $j \in \mathcal{C}'$ und alle $k \in \mathcal{R}_j$ erfüllt, wenn Vorgang i eingeplant wird ($\mathcal{C}' := \mathcal{C} \cup \{i\}$), Vorgänge ausgeplant werden oder die Anzahl der eingeplanten Vorgänge unverändert bleibt ($S'_i > S_i$). □

Satz 5.5. *Algorithmus 5.6 generiert alle aktiven Schedules, d. h.,* $\Phi \supseteq \mathcal{AS}$.

Beweis. Analog zum Beweis von Satz 5.4 unter Anwendung von Lemma 5.6 anstelle von Lemma 5.4. □

5.3.2 Bestimmung der Einplanungszeitpunkte

In diesem Abschnitt wird gezeigt, wie die Einschränkung T_i einer beliebigen Menge Θ_i, die in einer Iteration von Algorithmus 5.5 oder 5.6 für einen Vorgang $i \in \bar{\mathcal{C}}$ bestimmt wurde, berechnet werden kann. Im Folgenden werden zwei unterschiedliche Verfahren zur Berechnung der Einschränkung

$$T_i := \{\tau \in \Theta_i \mid \nexists \tau' \in [0, \tau[\cap \Theta_i : r_{ik}^u(\tau) \geq r_{ik}^u(\tau') \text{ für alle } k \in \mathcal{R}_i\}$$

vorgestellt. Analog zu den Startzeitbeschränkungen werden die Begriffe Startzeitintervall und Startzeitunterbrechung für Θ_i und T_i definiert, wobei beide Mengen gleichermaßen durch Listen, bestehend aus den Start- und Endzeitpunkten aller Startzeitintervalle, abgespeichert werden.

Das erste Verfahren zur Bestimmung der Einschränkung T_i wird in Algorithmus 5.7 beschrieben. Zur Vereinfachung der Darstellung wird $\min \emptyset := \infty$ vorausgesetzt. Im Verlauf des Verfahrens stellt die Variable t den in der aktuellen Iteration betrachteten Startzeitpunkt von Vorgang $i \in V$ aus der Menge Θ_i dar, wohingegen die boolesche Variable *element* angibt, ob der Startzeitpunkt t der Menge T_i am Ende einer Iteration hinzugefügt wird (*element* = *true*) oder nicht (*element* = *false*). Das Verfahren startet mit $T_i := \emptyset$ und überprüft in jeder Iteration für einen Startzeitpunkt $t \in \Theta_i$, ob in der Menge T_i ein Zeitpunkt τ existiert, der die Bedingungen $r_{ik}^u(t) \geq r_{ik}^u(\tau)$ für alle $k \in \mathcal{R}_i$ erfüllt. Dazu wird für jeden Zeitpunkt $\tau \in T_i$ überprüft, ob mindestens eine Ressource $k \in \mathcal{R}_i$ existiert, für die $r_{ik}^u(t) < r_{ik}^u(\tau)$ gilt. Falls diese Bedingung für alle Zeitpunkte in T_i erfüllt

Algorithmus 5.7: Einschränkung der Einplanungszeitpunkte (V1)

Input: Menge der Einplanungszeitpunkte Θ_i
Output: Eingeschränkte Menge der Einplanungszeitpunkte T_i

```
1  T_i := ∅,   t := min Θ_i,   element := true
2  while t < ∞ do
3  |  forall τ ∈ T_i do
4  |  |  element := false
5  |  |  forall k ∈ R_i do
6  |  |  |  if r^u_ik(t) < r^u_ik(τ) then
7  |  |  |  |  element := true
8  |  |  |  |  break
9  |  |  if element = false then
10 |  |  |  break
11 |  if element = true then
12 |  |  T_i := T_i ∪ {t}
13 |  t := min{τ ∈ Θ_i | τ > t}
14 return T_i
```

ist, woraus folgt, dass kein Zeitpunkt $\tau \in T_i$ mit $r^u_{ik}(t) \geq r^u_{ik}(\tau)$ für alle $k \in \mathcal{R}_i$ existiert, wird t der Menge T_i hinzugefügt. Da alle Zeitpunkte $t \in \Theta_i$ in einer Reihenfolge steigender Werte durchlaufen werden, gilt für jeden Zeitpunkt t, der der Menge T_i hinzugefügt wird, dass auch kein kleinerer Zeitpunkt $\tau \in \Theta_i$ mit $r^u_{ik}(t) \geq r^u_{ik}(\tau)$ für alle $k \in \mathcal{R}_i$ existiert. Dieser Zusammenhang kann daraus abgeleitet werden, dass für jeden Zeitpunkt $\tau \in \Theta_i$, der im Verlauf des Verfahrens nicht der Menge T_i hinzugefügt wurde, mindestens ein kleinerer Zeitpunkt $\tau' \in T_i$ mit $r^u_{ik}(\tau') \leq r^u_{ik}(\tau)$ für alle $k \in \mathcal{R}_i$ existiert. Da weiterhin für jeden Zeitpunkt $t \in \Theta_i$, der nicht der Menge T_i hinzugefügt wird, mindestens ein kleinerer Zeitpunkt $\tau \in T_i$ ($\tau \in \Theta_i$) mit $r^u_{ik}(t) \geq r^u_{ik}(\tau)$ für alle $k \in \mathcal{R}_i$ vorliegt, ist Algorithmus 5.7 korrekt bzw. gibt die Einschränkung T_i von Θ_i zurück.

Satz 5.6. *Algorithmus 5.7 hat eine Zeitkomplexität von* $\mathcal{O}(\bar{d}^2|\mathcal{R}|)$.

Beweis. Zunächst werden in jeder Iteration die Ressourcenbelegungen für den Zeitpunkt $t \in \Theta_i$ mit den Ressourcenbelegungen von höchstens $\mathcal{O}(\bar{d})$ weiteren Zeitpunkten verglichen. Da für jeden Vergleich maximal $|\mathcal{R}|$ Ressourcen betrachtet werden, sind höchstens $\mathcal{O}(\bar{d}|\mathcal{R}|)$ Operationen für die Gegenüberstellung der Ressourcenbelegungen von Zeitpunkt t und allen kleineren Zeitpunkten erforderlich. Durch die Verwendung der Listen $[r^u_{ik}(t)]$ zur Bestimmung der Ressourcenbelegungen für jeden Zeitpunkt $t \in \Theta_i$ und durch das Abspeichern der Ressourcenbelegungen für alle Zeitpunkte, die der Menge T_i hinzugefügt wurden, ergibt sich eine Zeitkomplexität von $\mathcal{O}(\bar{d}|\mathcal{R}| + \mathcal{I})$ für jede Iteration. Aus $\bar{d} \geq \mathcal{I}_k$

bzw. $\bar{d}|\mathcal{R}| \geq \mathcal{I}$ folgt schließlich eine Zeitkomplexität von $\mathcal{O}(\bar{d}^2|\mathcal{R}|)$ über alle Iterationen von Algorithmus 5.7. □

Im Gegensatz zu Algorithmus 5.7, der im ungünstigsten Fall die Ressourcenbelegungen für jeden Zeitpunkt $t \in \Theta_i$ mit den Ressourcenbelegungen aller kleineren Zeitpunkte in Θ_i vergleicht, werden im zweiten Verfahren Eigenschaften der Verläufe der Ressourcenbelegungen für die Berechnung von T_i ausgenutzt. Zur Vereinfachung wird im Folgenden $\Delta^u_{ik}(t) := r^u_{ik}(t+1) - r^u_{ik}(t)$ definiert, um die Veränderung der Belegung einer Ressource $k \in \mathcal{R}$ durch einen Vorgang $i \in V$ zu beschreiben, falls der Startzeitpunkt t von Vorgang i um eine Zeiteinheit erhöht wird. Aus den Beschreibungen zur Generierung der Listen $[r^u_{ik}(t)]$ in Abschnitt 4.2 folgt zunächst, dass der Planungshorizont für jeden Vorgang $i \in V$ und jede Ressource $k \in \mathcal{R}_i$ in $\mathcal{O}(\mathcal{I}_k)$ Zeitintervalle $[a, b]$ mit $\Delta^u_{ik}(t) = \Delta^u_{ik}(t+1)$ für alle $t \in [a, b[\cap \mathbb{Z}$ zerlegt werden kann. Im weiteren Verlauf wird davon gesprochen, dass der Belegungsverlauf einer Ressource $k \in \mathcal{R}$ auf einem Zeitintervall $[a, b]$ mit $a, b \in \mathcal{H}$ für einen Vorgang $i \in V$ konstant ist, falls $\Delta^u_{ik}(t) = \Delta^u_{ik}(t+1)$ für alle $t \in [a, b[\cap \mathbb{Z}$ gilt. Für jeden Vorgang $i \in V$ kann der Planungshorizont in eine polynomiell beschränkte Anzahl $\mathcal{O}(\mathcal{I})$ an Zeitintervallen mit konstanten Belegungsverläufen aller Ressourcen $k \in \mathcal{R}_i$ zerlegt werden. Abbildung 5.1 zeigt ein Beispiel für konstante Belegungsverläufe auf einem Zeitintervall $[a, b]$ für einen realen Vorgang $i \in V^r$. Als Grundlage für das zweite Verfahren zur Berechnung von T_i werden in Lemma 5.7 zunächst Bedingungen aufgestellt, die jeder Zeitpunkt in T_i in Bezug auf ein Zeitintervall mit konstanten Belegungsverläufen aller Ressourcen erfüllen muss.

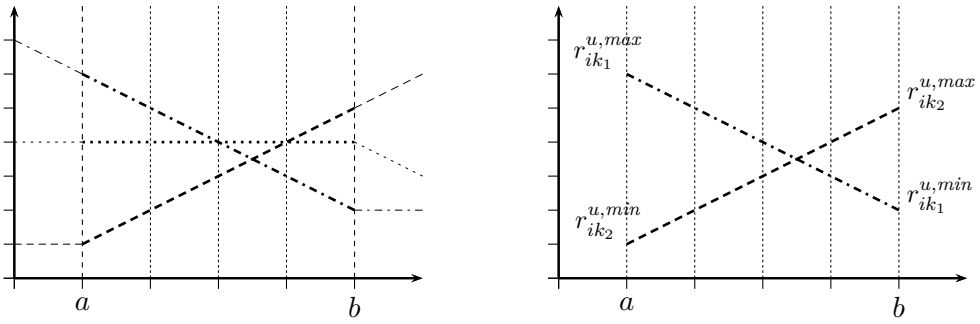

Abbildung 5.1: Zeitintervall mit konstanten Belegungsverläufen

Lemma 5.7. *Sei Θ_i die Menge aller Einplanungszeitpunkte, die für einen Vorgang $i \in \bar{\mathcal{C}}$ in einer Iteration von Algorithmus 5.5 oder 5.6 bestimmt wurde. Weiterhin sei eine Menge $\{a, a+1, \ldots, b\} \subseteq \Theta_i$ mit $a < b$ und $\Delta^u_{ik}(\tau') = \Delta^u_{ik}(a)$ für alle $\tau' \in [a, b[\cap \mathbb{Z}$ und alle $k \in \mathcal{R}_i$ gegeben. Dann erfüllt jeder Zeitpunkt $\tau \in \Theta_i$ genau dann die Bedingung*

$$\nexists \tau' \in \{a, a+1, \ldots, b\} : r^u_{ik}(\tau) \geq r^u_{ik}(\tau') \text{ für alle } k \in \mathcal{R}_i, \tag{5.2}$$

wenn mindestens eine der Bedingungen

$$\exists k \in \mathcal{R}_i : r^u_{ik}(\tau) < r^{u,min}_{ik} \tag{5.3}$$

$$\exists (k_1, k_2) \in \mathcal{R}^- \times \mathcal{R}^+ : r^u_{ik_1}(\tau) + r^u_{ik_2}(\tau) < r^{u,min}_{ik_1} + r^{u,min}_{ik_2} + b - a \tag{5.4}$$

mit $r^{u,min}_{ik} := \min\{r^u_{ik}(\tau) \mid \tau \in [a,b] \cap \mathbb{Z}\}$, $\mathcal{R}^- := \{k \in \mathcal{R}_i \mid \Delta^u_{ik}(a) < 0\}$ *und* $\mathcal{R}^+ := \{k \in \mathcal{R}_i \mid \Delta^u_{ik}(a) > 0\}$ *zutrifft.*

Beweis. Zunächst wird gezeigt, dass die Bedingung (5.2) für jeden Zeitpunkt $\tau \in \Theta_i$ erfüllt ist, falls mindestens eine der beiden Bedingungen (5.3) oder (5.4) gilt. Es wird zunächst angenommen, dass die Bedingung (5.3) erfüllt ist. Dann kann (5.2) direkt daraus abgeleitet werden, dass mindestens eine Ressource $k \in \mathcal{R}_i$ mit $r^u_{ik}(\tau) < r^u_{ik}(\tau')$ für alle $\tau' \in [a,b] \cap \mathbb{Z}$ existiert. Als Nächstes wird vorausgesetzt, dass (5.4) erfüllt ist, während (5.3) nicht zutrifft. Es folgt $r^u_{ik_1}(\tau)+r^u_{ik_2}(\tau) < r^{u,max}_{ik_1}+r^{u,min}_{ik_2}$ mit $r^{u,max}_{ik_1} := \max\{r^u_{ik_1}(\tau)|\tau \in [a,b] \cap \mathbb{Z}\} = r^{u,min}_{ik_1}+b-a$ für mindestens ein Paar $(k_1,k_2) \in \mathcal{R}^- \times \mathcal{R}^+$ (s. Abbildung 5.1). In Verbindung mit $r^u_{ik_2}(\tau) \geq r^{u,min}_{ik_2}$ ergibt sich schließlich $r^u_{ik_1}(\tau) < r^{u,max}_{ik_1}$, woraus $\exists \tau' \in [a+1,b] \cap \mathbb{Z}$ mit $r^u_{ik_1}(\tau) = r^u_{ik_1}(\tau')$ folgt. Da $r^u_{ik_1}(\tau'') + r^u_{ik_2}(\tau'') = c_{k_1k_2} := r^{u,max}_{ik_1} + r^{u,min}_{ik_2}$ für alle $\tau'' \in [a,b] \cap \mathbb{Z}$ gilt (s. Abbildung 5.1), ergibt sich aus $r^u_{ik_1}(\tau) = r^u_{ik_1}(\tau') = c_{k_1k_2} - r^u_{ik_2}(\tau')$ zusammen mit (5.4) die Bedingung $r^u_{ik_2}(\tau) < r^u_{ik_2}(\tau')$. Abschließend kann abgeleitet werden, dass die Bedingung (5.2) erfüllt ist, da $r^u_{ik_1}(\tau) = r^u_{ik_1}(\tau') < r^u_{ik_1}(\tau'')$ für alle $\tau'' \in [a,\tau'[\cap \mathbb{Z}$ und $r^u_{ik_2}(\tau) < r^u_{ik_2}(\tau') \leq r^u_{ik_2}(\tau'')$ für alle $\tau'' \in [\tau',b] \cap \mathbb{Z}$ gilt.

Im nächsten Schritt wird gezeigt, dass die Bedingung (5.2) nicht erfüllt ist, falls weder die Bedingung (5.3) noch (5.4) zutrifft. Dazu wird vorausgesetzt, dass die Bedingungen (5.3) und (5.4) nicht erfüllt sind. Zunächst wird angenommen, dass $r^u_{ik1}(\tau) \geq r^{u,max}_{ik1}$ für alle $k_1 \in \mathcal{R}^-$ gegeben ist. Dann wird die Bedingung (5.2) nicht erfüllt, da $r^u_{ik}(\tau) \geq r^u_{ik}(a)$ für alle $k \in \mathcal{R}_i$ gilt. Entsprechend wird nur noch der Fall betrachtet, dass mindestens eine Ressource $k_1 \in \mathcal{R}^-$ mit $r^u_{ik_1}(\tau) < r^{u,max}_{ik_1}$ vorliegt. Da in diesem Fall mindestens ein Zeitpunkt $\tau' \in [a+1,b] \cap \mathbb{Z}$ mit $r^u_{ik_1}(\tau) = r^u_{ik_1}(\tau')$ existiert, ergibt sich analog zum ersten Teil des Beweises $r^u_{ik_2}(\tau) \geq r^u_{ik_2}(\tau')$ für jede Ressource $k_2 \in \mathcal{R}^+$. Im Folgenden sei der Zeitpunkt $\bar{\tau} := \max\{\tau' \in [a+1,b] \cap \mathbb{Z} \mid \exists k \in \mathcal{R}^- : r^u_{ik}(\tau) = r^u_{ik}(\tau')\}$ gegeben, für den $r^u_{ik_2}(\tau) \geq r^u_{ik_2}(\bar{\tau})$ für alle $k_2 \in \mathcal{R}^+$ und $r^u_{ik_1}(\tau) \geq r^u_{ik_1}(\bar{\tau})$ für alle $k_1 \in \mathcal{R}^-$ direkt abgeleitet werden kann. Abschließend folgt aus $r^u_{ik}(\tau) \geq r^{u,min}_{ik}$ für alle $k \in \mathcal{R}_i$, dass die Bedingung (5.2) für den Zeitpunkt τ nicht zutrifft, da $r^u_{ik}(\tau) \geq r^u_{ik}(\bar{\tau})$ für alle $k \in \mathcal{R}_i$ gilt. □

Im Folgenden wird das zweite Verfahren zur Berechnung der Einschränkung T_i von Θ_i beschrieben, das in Algorithmus 5.8 skizziert wird. Wie auch das erste Verfahren startet

der Algorithmus 5.8 mit einer leeren Menge T_i, die im Verlauf des Verfahrens schrittweise ergänzt wird. In jeder Iteration wird eine Teilmenge Θ von Θ_i betrachtet, die zu Beginn des Verfahrens der Menge Θ_i entspricht und schrittweise reduziert wird. Im Folgenden wird zur Abgrenzung zu einer Startzeitbeschränkung W_i der Endzeitpunkt eines Startzeitintervalls in Θ_i, dem ein Zeitpunkt τ zugeordnet ist, durch $e^{\Theta_i}(\tau)$ angegeben. Weiterhin wird zur Abgrenzung der Bezeichner $\mathcal{B}^{\Theta_i}$ für die Anzahl der Startzeitunterbrechungen in Θ_i verwendet. In jeder Iteration von Algorithmus 5.8 werden die Variablen a und b bestimmt, die jeweils den Start- und Endzeitpunkt eines Intervalls $[a, b]$ innerhalb eines Startzeitintervalls von Θ_i mit $\{a, a+1, \ldots, b\} \subseteq \Theta_i$ angeben. Zunächst wird zu Beginn einer Iteration die Variable a auf den Startzeitpunkt des ersten Startzeitintervalls in Θ gesetzt. Falls der Endzeitpunkt des ersten Startzeitintervalls in Θ größer als der Startzeitpunkt ist ($a \neq e^{\Theta_i}(a)$), werden alle Ressourcen $k \in \mathcal{R}_i$ mit einem fallenden Belegungsverlauf $\Delta^u_{ik}(a) < 0$ bestimmt ($\mathcal{R}^-$). Für den Fall, dass $\mathcal{R}^- \neq \emptyset$ gilt, werden ebenso alle Ressourcen $k \in \mathcal{R}_i$ mit einem steigenden Belegungsverlauf $\Delta^u_{ik}(a) > 0$ ermittelt ($\mathcal{R}^+$), gefolgt von der Berechnung des größtmöglichen Werts b_{ik} für jede Ressource $k \in \mathcal{R}_i$, der einen konstanten Belegungsverlauf auf dem Zeitintervall $[a, b_{ik}]$ für die jeweilige Ressource sicherstellt. Entsprechend wird durch $b := \min\left(\min_{k\in\mathcal{R}_i} b_{ik}, e^{\Theta_i}(a)\right)$ ein Zeitintervall $[a, b] \cap \mathbb{Z} \subseteq \Theta_i$ mit einem konstanten Belegungsverlauf über alle Ressourcen $k \in \mathcal{R}_i$ auf $[a, b]$ bestimmt. Für den Fall, dass dagegen $a = e^{\Theta_i}(a)$ oder $\mathcal{R}^- = \emptyset$ gilt, wird stattdessen im weiteren Verlauf das Zeitintervall $[a, b]$ mit $b := a$ betrachtet. Nach der Berechnung der Mindestbelegungen $r^{u,min}_{ik} := \min\{r^u_{ik}(\tau) \,|\, \tau \in [a, b] \cap \mathbb{Z}\}$ auf dem Zeitintervall $[a, b]$ für alle Ressourcen $k \in \mathcal{R}_i$ werden die Zeitpunkte $\{a, a+1, \ldots, b\} \cap \Theta$ zunächst der Menge T_i hinzugefügt und anschließend aus der Menge Θ entfernt. Im nächsten Schritt wird weiterhin jeder Zeitpunkt t aus der Menge Θ entfernt, für den mindestens ein Zeitpunkt $\tau \in \{a, a+1, \ldots, b\}$ mit $r^u_{ik}(t) \geq r^u_{ik}(\tau)$ für alle $k \in \mathcal{R}_i$ existiert, sodass der Zeitpunkt t für die Erweiterung der Einschränkung T_i nicht infrage kommt. Die entsprechenden Operationen, die auf Lemma 5.7 basieren, werden in den Zeilen 14 bis 18 ausgeführt. Das Verfahren wird solange fortgesetzt, bis kein Zeitpunkt mehr in der Menge Θ vorliegt. Nach Ablauf des Verfahrens gibt Algorithmus 5.8 schließlich die Einschränkung T_i von Θ_i zurück.

Satz 5.7. *Algorithmus 5.8 bestimmt die Einschränkung T_i einer Menge $\Theta_i \subseteq \mathcal{H}$ mit einer Zeitkomplexität von $\mathcal{O}(|\mathcal{R}|^2(\mathcal{B}^{\Theta_i} + \mathcal{I})(\mathcal{B}^{\Theta_i} + |\mathcal{R}|\mathcal{I}))$.*

Beweis. Für den Beweis der Korrektheit von Algorithmus 5.8 wird zunächst mit Hilfe vollständiger Induktion gezeigt, dass

$$\Theta = \Theta^a_i := \{\tau \in \Theta_i \mid \tau \geq a \wedge \nexists \tau' \in [0, a[\cap \Theta_i : r^u_{ik}(\tau) \geq r^u_{ik}(\tau') \text{ für alle } k \in \mathcal{R}_i\}$$

Algorithmus 5.8: Einschränkung der Einplanungszeitpunkte (V2)

Input: Menge der Einplanungszeitpunkte Θ_i
Output: Eingeschränkte Menge der Einplanungszeitpunkte T_i

1 $T_i := \emptyset, \qquad \Theta := \Theta_i$
2 **while** $\Theta \neq \emptyset$ **do**
3 $\quad a := \min \Theta, \qquad b := a, \qquad \mathcal{R}^- := \emptyset, \qquad \mathcal{R}^+ := \emptyset$
4 $\quad$ **if** $a \neq e^{\Theta_i}(a)$ **then**
5 $\quad\quad \mathcal{R}^- := \{k \in \mathcal{R}_i \mid \Delta^u_{ik}(a) < 0\}$
6 $\quad\quad$ **if** $\mathcal{R}^- \neq \emptyset$ **then**
7 $\quad\quad\quad \mathcal{R}^+ := \{k \in \mathcal{R}_i \mid \Delta^u_{ik}(a) > 0\}$
8 $\quad\quad\quad$ **forall** $k \in \mathcal{R}_i$ **do**
9 $\quad\quad\quad\quad b_{ik} := \max\{\tau \in \mathcal{H} \mid \Delta^u_{ik}(\tau') = \Delta^u_{ik}(a) \text{ für alle } \tau' \in [a, \tau[\cap \mathbb{Z}\}$
10 $\quad\quad\quad b := \min\left(\min_{k \in \mathcal{R}_i} b_{ik}, e^{\Theta_i}(a)\right)$
11 $\quad$ **forall** $k \in \mathcal{R}_i$ **do**
12 $\quad\quad r^{u,min}_{ik} := \min\{r^u_{ik}(\tau) \mid \tau \in [a, b] \cap \mathbb{Z}\}$
13 $\quad T_i := T_i \cup ([a, b] \cap \Theta), \qquad \Theta := \Theta \setminus \{a, a+1, \ldots, b\}, \qquad \Theta' := \emptyset$
14 $\quad$ **forall** $k \in \mathcal{R}_i : r^{u,min}_{ik} > 0$ **do**
15 $\quad\quad \Theta' := \Theta' \cup \{\tau \in \Theta \mid r^u_{ik}(\tau) < r^{u,min}_{ik}\}$
16 $\quad$ **forall** $(k_1, k_2) \in \mathcal{R}^- \times \mathcal{R}^+$ **do**
17 $\quad\quad c := r^{u,min}_{ik_1} + r^{u,min}_{ik_2} + b - a$
18 $\quad\quad \Theta' := \Theta' \cup \{\tau \in \Theta \mid r^u_{ik_1}(\tau) + r^u_{ik_2}(\tau) < c\}$
19 $\quad \Theta := \Theta'$
20 **return** T_i

am Anfang jeder Iteration gilt. Der Induktionsanfang ist zunächst durch die erste Iteration gegeben. Weiterhin wird gezeigt, dass $\Theta = \Theta_i^{b+1}$ am Ende jeder Iteration gilt, falls die Induktionsannahme erfüllt ist, woraus sich $\Theta_i^{b+1} = \Theta_i^{\alpha}$ mit $\alpha = \min \Theta_i^{b+1}$ für eine direkt nachfolgende Iteration ergibt ($\Theta_i^{b+1} \neq \emptyset$). Betrachtet wird dazu die Aktualisierung von Θ durch die Operationen in den Zeilen 13 bis 19. Für den Fall $a < b$ kann $\Theta = \Theta_i^{b+1}$ am Ende einer Iteration direkt aus Lemma 5.7 abgeleitet werden, wohingegen die gleiche Aussage für $a = b$ gilt. Basierend auf $\Theta = \Theta_i^a$ zu Beginn jeder Iteration kann schließlich gezeigt werden, dass jeder Zeitpunkt $t \in [a, b] \cap \Theta$, der in jeder Iteration der Einschränkung T_i hinzugefügt wird (Zeile 13), auch tatsächlich ein Element von T_i ist. Während für den Fall $a = b$ der Zusammenhang direkt aus der Definition von T_i folgt, kann im Fall $a < b$ die Korrektheit dadurch gezeigt werden, dass mindestens eine Ressource $k \in \mathcal{R}^-$ vorliegt, sodass die Bedingung $r^u_{ik}(\tau') < r^u_{ik}(\tau)$ für jedes Paar $(\tau, \tau') \in \{a, a+1, \ldots, b\}^2$ mit $\tau < \tau'$ erfüllt ist. Da weiterhin kein Zeitpunkt $t \in [b+1, \alpha[\cap \Theta_i$ mit $\alpha = \min \Theta_i^{b+1}$ für den Fall $\Theta_i^{b+1} \neq \emptyset$ Element der Einschränkung T_i ist, folgt schließlich die Korrektheit des Verfahrens.

Um die Zeitkomplexität zu bestimmen, wird angenommen, dass Θ_λ der Menge Θ zu Beginn einer Iteration λ von Algorithmus 5.8 entspricht und $\mathcal{B}^{\Theta_\lambda}$ die Anzahl der Startzeitunterbrechungen in Θ_λ angibt. Zunächst werden die Operationen in den Zeilen 14 bis 18 betrachtet. Für beide Schleifendurchläufe können jeweils Zeitkomplexitäten von $\mathcal{O}(|\mathcal{R}|\mathcal{B}^{\Theta_\lambda}+\mathcal{I})$ und $\mathcal{O}(|\mathcal{R}|^2\mathcal{B}^{\Theta_\lambda}+|\mathcal{R}|\mathcal{I})$ durch die Verwendung der Listen $[r_{ik}^u(t)]$ und das Abspeichern von Θ durch die Start- und Endzeitpunkte aller enthaltenen Startzeitintervalle abgeleitet werden. Weiterhin ergibt sich eine maximale Anzahl an $\mathcal{O}(\mathcal{B}^{\Theta_i}+\mathcal{I})$ Iterationen für Algorithmus 5.8, da der Zeitpunkt a in zwei aufeinanderfolgenden Iterationen entweder einem anderen Startzeitintervall von Θ_i zugeordnet wird oder mindestens einen Zeitpunkt in einer der Listen $[r_{ik}^u(t)]$ überspringt. Es ist dabei zu beachten, dass Zeitpunkt a auch im Fall $a \neq e^{\Theta_i}(a)$ und $\mathcal{R}^- = \emptyset$ mindestens einen Zeitpunkt in einer der Listen $[r_{ik}^u(t)]$ überspringt, da Startzeitpunkt $\tau = \min \Theta_i^{b+1} - 1$ die Bedingung $\Delta_{ik}^u(\tau) < 0$ für mindestens eine Ressource $k \in \mathcal{R}_i$ erfüllt. Aufgrund der streng monotonen Zunahme von Zeitpunkt a über alle Iterationen von Algorithmus 5.8 kann $e^{\Theta_i}(a)$ über alle Iterationen dadurch bestimmt werden, dass die Start- und Endzeitpunkte aller Startzeitintervalle in Θ_i höchstens ein Mal durchlaufen werden. Das Gleiche gilt ebenso für alle abgespeicherten Zeitpunkte in den Listen $[r_{ik}^u(t)]$ zur Bestimmung von $\mathcal{R}^-$, $\mathcal{R}^+$, den Werten b_{ik} und der Mindestbelegung $r_{ik}^{u,min}$ auf dem Intervall $[a,b]$ für jede Ressource $k \in \mathcal{R}_i$. Zusammenfassend ergibt sich schließlich eine Zeitkomplexität von $\mathcal{O}(|\mathcal{R}|^2\mathcal{B}^{\Theta_\lambda}+|\mathcal{R}|\mathcal{I})$ für jede Iteration von Algorithmus 5.8, wobei T_i mit einer Zeitkomplexität von $\mathcal{O}(\mathcal{B}^{\Theta_\lambda})$ in jeder Iteration aktualisiert wird (Zeile 13). Unter Berücksichtigung, dass der Menge $\Theta := \Theta_i$ zu Beginn des Verfahrens durch die Operationen in den Zeilen 14 bis 18 höchstens $\mathcal{O}(|\mathcal{R}|\mathcal{I})$ zusätzliche Startzeitunterbrechungen über alle Iterationen hinzugefügt werden, ist die maximale Anzahl an Startzeitunterbrechungen in Θ_λ in jeder Iteration durch $\mathcal{O}(\mathcal{B}^{\Theta_i}+|\mathcal{R}|\mathcal{I})$ beschränkt. Schließlich folgt eine Zeitkomplexität von $\mathcal{O}(|\mathcal{R}|^2(\mathcal{B}^{\Theta_i}+\mathcal{I})(\mathcal{B}^{\Theta_i}+|\mathcal{R}|\mathcal{I}))$ für den Algorithmus 5.8. □

5.3.3 Techniken zur Redundanzvermeidung

Im Folgenden werden zwei verschiedene Techniken zur Redundanzvermeidung im Enumerationsverlauf von Algorithmus 5.6 vorgestellt. Beide Techniken basieren auf einer Reduktion der möglichen Einplanungszeitpunkte der Vorgänge in jedem Verzweigungsschritt des konstruktionsbasierten Enumerationsschemas. Im ersten Teil dieses Abschnitts werden zunächst beide Techniken zur Redundanzvermeidung einzeln betrachtet. Im Anschluss daran wird die gemeinsame Anwendung beider Techniken untersucht, wobei gezeigt werden kann, dass dadurch eine redundanzfreie Enumeration erreicht wird.

Für die erste Technik werden in jedem Verzweigungsschritt des Enumerationsschemas die Ressourcenbelegungen durch den ausgewählten Vorgang und dem ihm zugeordneten Zeitpunkt als untere Schrankenwerte für die Einplanungszeitpunkte im weiteren Enumerationsverlauf festgelegt. Dazu werden für die sogenannte Usage-Preserving-Technik (UPT) in jedem Verzweigungsschritt für jeden direkten Nachfolgerknoten nur die Startzeitpunkte des Vorgangs $i \in \bar{\mathcal{C}}$ in der Startzeitbeschränkung W_i' weiterhin betrachtet, deren induzierte Ressourcenbelegungen über alle Ressourcen $k \in \mathcal{R}_i$ nicht kleiner sind als die des gewählten Einplanungszeitpunkts $t \in T_i$. Zur Anwendung der UPT wird der Verzweigungsschritt des Enumerationsschemas wie folgt angepasst. Nachdem Vorgang $i \in \bar{\mathcal{C}}$ der Zeitpunkt $t \in T_i$ als frühestmöglicher Startzeitpunkt durch $S_i' := t$ in Zeile 15 von Algorithmus 5.6 zugeordnet wurde, werden $W' := W$ und $W_i' := \{\tau \in W_i' | r_{ik}^u(\tau) \geq r_{ik}^u(t)$ für alle $k \in \mathcal{R}_i\}$ gesetzt. Weiterhin werden die darauffolgenden Operationen in Zeile 15 auf der Startzeitbeschränkung W' an Stelle von W ausgeführt. Um zu zeigen, dass Algorithmus 5.6 durch die Anwendung der UPT auch weiterhin alle aktiven Schedules generiert bzw. korrekt ist, wird im Folgenden gezeigt, dass Lemma 5.6 auch dann gilt, falls die UPT ausgeführt wird. Dazu wird der Beweis von Lemma 5.6 dahingehend erweitert, dass die angepassten Operationen für die UPT in Zeile 15 berücksichtigt werden. Gegeben sei der Zeitpunkt $t \in T_i$ mit $t = \min\{\tau \in \Theta_i \mid r_{ik}^u(\tau) \leq r_{ik}^u(S_i^f)$ für alle $k \in \mathcal{R}_i\}$, sodass $t \leq S_i^f$ und $r_{ik}^u(t) \leq r_{ik}^u(S_i^f)$ für alle $k \in \mathcal{R}_i$ gilt. Dann folgt zunächst $S^f \in \mathcal{S}_T(W')$, nachdem $W' := W$ initialisiert und $W_i' := \{\tau \in W_i' \mid r_{ik}^u(\tau) \geq r_{ik}^u(t)$ für alle $k \in \mathcal{R}_i\}$ gesetzt wurde. Analog zum Beweis von Lemma 5.6 können $S' := \min \widetilde{\mathcal{S}}_T(W', i, t) \leq S^f$ und $S^f \in \mathcal{S}_T(W')$ mit $W' := (W_j' \setminus [0, S_j'[)_{j \in V}$ abgeleitet werden, wohingegen der restliche Beweis unverändert übernommen werden kann. Zusammenfassend gilt Lemma 5.6 auch dann für den Algorithmus 5.6, wenn die UPT ausgeführt wird, wobei die Korrektheit des Verfahrens direkt daraus folgt (vgl. Abschnitt 5.3.1). Da durch die Anwendung der UPT in Algorithmus 5.6 für jeden Enumerationsknoten $(\mathcal{C}, S, W)$ und jeden Schedule $S' \in \mathcal{S}_T(W)$ die Bedingungen $S \leq S'$ und $r_{jk}^u(S_j) \leq r_{jk}^u(S_j')$ für alle $j \in \mathcal{C}$ und alle $k \in \mathcal{R}_j$ erfüllt sind, folgt Korollar 5.1 direkt aus Lemma 5.6.

Korollar 5.1. *Es sei angenommen, dass Algorithmus 5.6 mit der UPT ausgeführt wird. Weiterhin sei $S^f \in \mathcal{S}$ ein beliebiger zulässiger Schedule und $(\mathcal{C}, S, W)$ ein Knoten im Enumerationsverlauf von Algorithmus 5.6 mit $\mathcal{C} \neq V$ und $S^f \in \mathcal{S}_T(W)$. Dann existiert mindestens ein direkter Nachfolgerknoten $(\mathcal{C}', S', W')$, der die Bedingung $S^f \in \mathcal{S}_T(W')$ erfüllt.*

Da $\mathcal{S}_T(W') \subseteq \mathcal{S}_T(W)$ für einen Knoten $(\mathcal{C}, S, W)$ und jeden seiner Nachfolgerknoten $(\mathcal{C}', S', W')$ gilt, ergibt sich aus Korollar 5.1, dass genau die Menge $\mathcal{S}_T(W)$ durch Knoten $(\mathcal{C}, S, W)$ und alle Nachfolgerknoten untersucht wird. Es ist dabei zu beachten, dass ohne

die UPT für einen Knoten $(\mathcal{C}, S, W)$ im Enumerationsverlauf auch Bereiche in $\mathcal{S}_T(W)$ untersucht werden könnten, die keine Teilmengen von $\{S' \in \mathcal{S}_T(W) \mid r^u_{jk}(S'_j) \geq r^u_{jk}(S_j)$ für alle $j \in \mathcal{C}$ und alle $k \in \mathcal{R}_j\}$ sind.

Die zweite Technik zur Redundanzvermeidung basiert auf der Betrachtung der Ressourcenbelegungen aller Zeitpunkte $\tau \in T_i$ eines ausgewählten Vorgangs $i \in \bar{\mathcal{C}}$ im Verzweigungsschritt von Algorithmus 5.6, die kleiner als der zugeordnete Zeitpunkt t sind. Für die Anwendung der zweiten Technik zur Redundanzvermeidung wird Zeile 15 von Algorithmus 5.6 analog zur UPT angepasst, wobei der einzige Unterschied darin besteht, dass $W'_i := \{\tau \in W'_i \mid \nexists \tau' \in [0, t[\cap T_i : r^u_{ik}(\tau) \geq r^u_{ik}(\tau')$ für alle $k \in \mathcal{R}_i\}$ gesetzt wird. Entsprechend wird durch die zweite Redundanzvermeidungstechnik Zeitpunkt τ aus W'_i entfernt, falls mindestens ein Zeitpunkt τ' in T_i vorliegt, der kleiner als der zugeordnete Zeitpunkt t ist und zugleich die Bedingungen $r^u_{ik}(\tau) \geq r^u_{ik}(\tau')$ für alle $k \in \mathcal{R}_i$ erfüllt. Da daraus folgt, dass für jeden Zeitpunkt $\tau \in W'_i$ mindestens eine Ressource $k \in \mathcal{R}_i$ mit $r^u_{ik}(\tau) < r^u_{ik}(\tau')$ für jeden Zeitpunkt $\tau' \in [0, t[\cap T_i$ existiert, wird das zweite Verfahren zur Redundanzvermeidung als Usage-Limitation-Technik (ULT) bezeichnet. Um zu zeigen, dass Lemma 5.6 auch weiterhin gilt, falls Algorithmus 5.6 mit der ULT ausgeführt wird, kann der Beweis von Lemma 5.6 wie für die UPT erweitert werden, wobei lediglich die Aktualisierung von W'_i ersetzt wird. Entsprechend ist es ausreichend zu zeigen, dass $S^f_i \in W'_i$ nach der Aktualisierung $W'_i := \{\tau \in W'_i \mid \nexists \tau' \in [0, t[\cap T_i : r^u_{ik}(\tau) \geq r^u_{ik}(\tau')$ für alle $k \in \mathcal{R}_i\}$ mit $t = \min\{\tau \in \Theta_i \mid r^u_{ik}(\tau) \leq r^u_{ik}(S^f_i)$ für alle $k \in \mathcal{R}_i\}$ gilt. Da $S^f_i \notin W'_i$ einen Widerspruch zur Annahme $t = \min\{\tau \in \Theta_i \mid r^u_{ik}(\tau) \leq r^u_{ik}(S^f_i)$ für alle $k \in \mathcal{R}_i\}$ ergeben würde, folgt schließlich, dass Lemma 5.6 auch dann gilt, wenn Algorithmus 5.6 mit der ULT ausgeführt wird. Die Korrektheit von Algorithmus 5.6 mit der ULT ergibt sich analog zu Abschnitt 5.3.1.

Abschließend wird die gemeinsame Anwendung beider Techniken zur Redundanzvermeidung untersucht. Dazu werden für einen beliebigen Knoten im Enumerationsverlauf von Algorithmus 5.6 zwei direkte Nachfolgerknoten $(\mathcal{C}', S', W')$ und $(\mathcal{C}'', S'', W'')$ betrachtet. Aus den Beschreibungen beider Redundanzvermeidungstechniken kann zunächst $W'_i \cap W''_i = \emptyset$ abgeleitet werden. Da daraus $\mathcal{S}_T(W') \cap \mathcal{S}_T(W'') = \emptyset$ folgt, ergibt sich schließlich Korollar 5.2 aus Korollar 5.1.

Korollar 5.2. *Es sei angenommen, dass Algorithmus 5.6 mit der UPT und der ULT ausgeführt wird. Weiterhin sei $S^f \in \mathcal{S}$ ein beliebiger zulässiger Schedule und $(\mathcal{C}, S, W)$ ein Knoten im Enumerationsverlauf von Algorithmus 5.6 mit $\mathcal{C} \neq V$ und $S^f \in \mathcal{S}_T(W)$. Dann existiert genau ein direkter Nachfolgerknoten $(\mathcal{C}', S', W')$, der die Bedingung $S^f \in \mathcal{S}_T(W')$ erfüllt.*

Unter Berücksichtigung, dass $\mathcal{S}_T(W') \subseteq \mathcal{S}_T(W)$ für jeden Knoten $(\mathcal{C}, S, W)$ und jeden seiner Nachfolgerknoten $(\mathcal{C}', S', W')$ gilt, ergibt sich aus Korollar 5.2, dass jeder zulässige Schedule durch Algorithmus 5.6 unter Anwendung der UPT und der ULT höchstens ein Mal generiert wird. Weiterhin wird ein redundanzfreier Enumerationsbaum erzeugt, sodass eine beliebige Teilmenge von $\mathcal{S}_T$ höchstens auf einem Weg des Enumerationsbaums untersucht wird.

5.3.4 Branch-and-Bound-Algorithmus

In diesem Abschnitt wird das konstruktionsbasierte Branch-and-Bound-Verfahren vorgestellt, das auf dem Enumerationsschema in Algorithmus 5.6 basiert. Der erste Teil dieses Abschnitts betrachtet zunächst die Suchstrategie des Verfahrens bzw. geht darauf ein, wie der Suchbaum im Enumerationsverlauf generiert wird. Im Anschluss daran wird beschrieben, wie das Enumerationsschema um untere Schrankenwerte, Konsistenztests und Techniken zur Redundanzvermeidung erweitert werden kann.

Zunächst werden zur Bestimmung einer Reihenfolge, in der alle bereits generierten aber noch nicht vollständig untersuchten Knoten im Enumerationsverlauf betrachtet werden (Traversierungsstrategie), die Tiefensuche (DFS) sowie die Scattered-Path-Suche (SPS) verwendet. Nachdem durch die Traversierungsstrategie der nächste zu untersuchende Knoten festgelegt wurde, bestimmt die Verzweigungsstrategie den Vorgang für den Verzweigungsschritt des Enumerationsschemas. Zunächst wird dazu die Menge aller einplanbaren Vorgänge $\mathcal{E} \subseteq \bar{\mathcal{C}}$ festgelegt. Die erste Alternative betrachtet für $\mathcal{E}$ alle aktuell nicht eingeplanten Vorgänge ($\bar{\mathcal{C}}$), d. h., es wird $\mathcal{E} := \bar{\mathcal{C}}$ gesetzt. Die zwei weiteren Alternativen, die beide auf einer strikten Halbordnung $\prec$ auf der Vorgangsmenge V basieren, schränken die Menge $\bar{\mathcal{C}}$ dagegen ein, wobei $\mathcal{E} \subseteq \{i \in \bar{\mathcal{C}} \mid \mathit{Pred}^{\prec}(i) \subseteq \mathcal{C}\}$ gilt mit $\mathit{Pred}^{\prec}(i) := \{h \in V \mid (h,i) \in \mathit{tr}(\prec)\}$ und $\mathit{tr}(\prec)$ als transitive Reduktion von $\prec$. Für die beiden Alternativen werden die Distanzordnung $\prec_{\mathrm{D}}$ und die Zyklenordnung $\prec_{\mathrm{C}}$, die in Franck et al. (2001b) und Neumann et al. (2003, Abschnitt 2.6) definiert werden, verwendet. Genaue Definitionen und Beschreibungen zu den Ordnungen können den angegebenen Literaturquellen entnommen werden. Nach der Bestimmung der Menge aller einplanbaren Vorgänge $\mathcal{E}$ wird schließlich ein Vorgang mit bestem Prioritätsregelwert π_i und kleinstem Index ausgewählt. Entsprechend wird der Vorgang für den Verzweigungsschritt durch

$$i := \min\{i' \in \mathcal{E} \mid \pi_{i'} = \operatorname*{ext}_{l \in \mathcal{E}} \pi_l\}$$

bestimmt, wobei $\mathrm{ext} \in \{\min, \max\}$ angibt, ob entweder kleinere (min) oder größere (max) Prioritätsregelwerte bevorzugt werden. Im Folgenden werden unterschiedliche Prioritäts-

regeln vorgestellt, die sich in Voruntersuchungen als vorteilhaft herausgestellt haben. Zunächst werden Prioritätsregeln betrachtet, die sich auf die Zeitrestriktionen zwischen den Vorgängen beziehen und in der Literatur zur Projektplanung bereits umfassend untersucht wurden (s. z. B. Kolisch, 1996; Franck et al., 2001b). Zu den entsprechenden Prioritätsregeln gehören unter anderem die LST-Regel (engl. Latest Start Time), die jedem Vorgang $i \in \mathcal{E}$ einen Prioritätsregelwert $\pi_i = LS_i$ zuordnet und die ST-Regel (engl. Slack Time) mit $\pi_i = LS_i - ES_i$. Für beide Prioritätsregeln wurden zusätzlich dynamische Varianten implementiert, die sowohl die Startzeitbeschränkungen als auch die obere Schranke UB im Enumerationsverlauf einbeziehen. Die dynamische Variante der LST-Regel (LSTd) setzt als Prioritätsregelwert $\pi_i = LS_i^{UB}(W)$ und die dynamische Variante der ST-Regel (STd)[3] $\pi_i = LS_i^{UB}(W) - ES_i(W)$ mit $LS_i^{UB}(W) := LS_i(W, n+1, UB-1)$. Zusätzlich wurde für die ST-Regel eine dynamische Variante untersucht, die durch $\pi_i = |W_i \cap [ES_i(W), LS_i^{UB}(W)]|$ die Anzahl der Startzeitpunkte in der Startzeitbeschränkung des jeweiligen Vorgangs berücksichtigt (STd^{I}). Weitere Prioritätsregeln, die untersucht wurden, sind die PF-Regel (engl. Path Following) mit $\pi_i = l(i)$ und $l(i)$ als maximale Anzahl der Knoten über alle längsten Wege in Netzplan N von Vorgang i zu $n+1$, die MRC-Regel (engl. Maximal Resource Consumption) mit $\pi_i = p_i \sum_{k \in \mathcal{R}_i} r_{ik}^d$ und die TS-Regel (engl. Total Successor) mit $\pi_i = |\{j \in V \mid (i,j) \in tc(\prec)\}|$ und $tc(\prec)$ als transitive Hülle von $\prec$. Im Gegensatz zu den bisherigen Prioritätsregeln werden im Folgenden Regeln vorgestellt, die die Eigenschaften der partiell erneuerbaren Ressourcen ausnutzen. Dazu wird die maximal mögliche Inanspruchnahme $p_i\, r_{ik}^d$ einer Ressource $k \in \mathcal{R}$ durch einen Vorgang $i \in \mathcal{E}$ im Verhältnis zur maximal verbleibenden Kapazität $\bar{R}_k$ mit

$$\bar{R}_k := R_k - r_k^c(S^{\mathcal{C}}) - \sum_{i \in \bar{\mathcal{C}}} r_{ik}^{c,min}(W, ES_i(W), LS_i^{UB}(W))$$

betrachtet. Es wurden die folgenden Prioritätsregeln untersucht. Die TMAR-Regel (engl. Total Maximal Additional Resource Consumption) mit

$$\pi_i = p_i \sum_{k \in \mathcal{R}_i : \bar{R}_k \neq 0} \frac{r_{ik}^d}{\bar{R}_k} + \sum_{k \in \mathcal{R}_i : \bar{R}_k = 0} 1$$

und die AMAR-Regel (engl. Average Maximal Additional Resource Consumption) mit $\pi_i = \pi_i' / |\mathcal{R}_i|$ und π_i' als Prioritätsregelwert der TMAR-Regel.

[3] Es ist zu beachten, dass die dynamische Variante der ST-Regel (STd) im Allgemeinen nicht mit der TF-Regel, die in Abschnitt 5.1.4 eingeführt wurde, übereinstimmt. Der Grund dafür ist, dass sich durch die Anwendung von Konsistenztests im konstruktionsbasierten Branch-and-Bound-Verfahren die Werte von S_i und $ES_i(W)$ in einem Enumerationsknoten unterscheiden können.

Nach der Bestimmung des Vorgangs für den Verzweigungsschritt wird durch die Generierungsstrategie zunächst festgelegt, wie viele direkte Nachfolgerknoten maximal generiert werden. Dabei können wie im relaxationsbasierten Branch-and-Bound-Verfahren entweder alle direkten Nachfolgerknoten auf einmal erzeugt werden (all) oder nur eine begrenzte maximale Anzahl (restr). Falls nur eine begrenzte Anzahl an direkten Nachfolgerknoten erzeugt wird, gibt die Generierungsstrategie weiterhin die Reihenfolge vor, in der die betrachteten Zeitpunkte in T_i durchlaufen werden. Die erste Alternative LT (engl. Lowest Time) betrachtet in jedem Verzweigungsschritt den kleinsten Zeitpunkt in T_i, der bislang noch nicht betrachtet wurde. Bei der anderen Alternative PV (engl. Priority Value) wird dagegen jedem Zeitpunkt t in T_i ein Prioritätsregelwert π_t zugeordnet und alle Zeitpunkte in T_i nach nichtfallenden Werten von π_t durchlaufen, wobei im Fall von gleichen Prioritätsregelwerten immer der kleinste Zeitpunkt gewählt wird. Im Folgenden wird eine Prioritätsregel für die Bestimmung einer Reihenfolge der Zeitpunkte in T_i beschrieben, die in Voruntersuchungen die besten Ergebnisse erzielen konnte. Der zugehörige Prioritätregelwert

$$\pi_t = \sum_{k \in \mathcal{R}_i} a_{ikt} + \frac{1}{4} \max_{k \in \mathcal{R}_i}(b_{ik})(t - ES_i(W))$$

mit

$$a_{ikt} := \begin{cases} r^c_{ik}(t)/\bar{R}_k, & \text{falls } \bar{R}_k \neq 0 \\ 1, & \text{sonst} \end{cases} \quad \text{und} \quad b_{ik} := \begin{cases} r^d_{ik}/\bar{R}_k, & \text{falls } \bar{R}_k \neq 0 \\ 1, & \text{sonst} \end{cases}$$

kann als Kombination aus einem Prioritätsregelwert, der sich an die TMAR-Regel anlehnt und einem Bestrafungsterm, der den Wert von π_t linear zum Zeitabstand zwischen Zeitpunkt t und $ES_i(W)$ erhöht, interpretiert werden.

Abschließend wird durch die Reihenfolgestrategie festgelegt, in welcher Reihenfolge alle in einem einzelnen Verzweigungsschritt generierten direkten Nachfolgerknoten im weiteren Enumerationsverlauf untersucht werden. In Voruntersuchungen konnte gezeigt werden, dass die besten Ergebnisse dadurch erzielt werden, dass alle generierten Knoten in einer Reihenfolge nach nichtfallenden Werten der unteren Schranken für die kürzeste Projektdauer untersucht werden. Entsprechend werden abweichend vom relaxationsbasierten Branch-and-Bound-Verfahren die generierten direkten Nachfolgerknoten in einem Verzweigungsschritt nicht zusätzlich durch Prioritätsregeln sortiert.

Im Folgenden wird das konstruktionsbasierte Branch-and-Bound-Verfahren beschrieben, das in Algorithmus 5.9 skizziert wird. Wie bereits für das relaxations- und partitionsbasierte Branch-and-Bound-Verfahren wird zur Vereinfachung der Darstellung in Algo-

Algorithmus 5.9: Konstruktionsbasierter Branch-and-Bound-Algorithmus

Input: RCPSP/max-π-Instanz
Output: Optimaler Schedule S^*

Bestimme die Distanzmatrix $D = (d_{ij})_{i,j \in V}$
if $\mathcal{S}_T = \emptyset$ **then** terminate ($\mathcal{S} = \emptyset$)
Setze $ES_i := d_{0i}$, $LS_i := -d_{i0}$ und $W_i := [ES_i, LS_i] \cap \mathbb{Z}$ für alle $i \in V$
Führe Preprocessing-Schritt auf W aus
if $\mathcal{S}_T(W) = \emptyset$ **then** terminate ($\mathcal{S} = \emptyset$)
$S := \min \mathcal{S}_T(W), \quad LB := S_{n+1}$
$\Omega := \{(\mathcal{C}, S, W, LB)\}, \quad UB := \bar{d} + 1$
while $\Omega \neq \emptyset$ **do**
  Entferne den obersten Knoten $(\mathcal{C}, S, W, LB)$ vom Stack Ω
  if $LB < UB$ **then**
    Führe Konsistenztests aus der Menge Γ^{D} oder Γ^{W} auf W aus
    if $\mathcal{S}_T^{UB}(W) \neq \emptyset$ **then**
      Initialisiere $\Lambda := \emptyset$ und wähle einen Vorgang $i \in \mathcal{E}$ gemäß der Verzweigungsstrategie
      $\Theta_i := \{\tau \in W_i \mid r_k^c(S^{\mathcal{C}}) + r_{ik}^c(\tau) \leq R_k \text{ für alle } k \in \mathcal{R}_i\}$
      Berechne T_i (Algorithmus 5.7 oder 5.8)
      while $T_i \neq \emptyset$ **do**
        Entferne einen Zeitpunkt t aus T_i gemäß der Generierungsstrategie
        $S_i' := t, \quad S' := \min \widetilde{\mathcal{S}}_T(W, i, S_i'), \quad W' := (W_j \setminus [0, S_j'[)_{j \in V}$
        Führe Konsistenztests aus der Menge Γ^{B} auf W' aus
        if $\mathcal{S}_T^{UB}(W') \neq \emptyset$ **then**
          $S' := \min \mathcal{S}_T(W')$
          if $\exists j \in \mathcal{C} : S_j' > S_j$ **then**
            $\mathcal{C}' := \mathcal{C} \setminus \{j \in \mathcal{C} \mid S_j' > S_j\}$
          else if $S_i' = t$ **then**
            $\mathcal{C}' := \mathcal{C} \cup \{i\}$
          if $\mathcal{C}' = V$ **then**
            $S^* := S', \quad UB := S_{n+1}^*, \quad T_i := T_i \setminus [t+1, \infty[$
          else
            Berechne die untere Schranke LB'
            if $LB' < UB$ **then**
              $\Lambda := \Lambda \cup \{(\mathcal{C}', S', W', LB')\}$
    Entferne alle Knoten aus der Liste Λ und lege sie gemäß der Reihenfolgestrategie auf den Stack Ω
if $UB = \bar{d} + 1$ **then** terminate ($\mathcal{S} = \emptyset$)
else return S^*

rithmus 5.9 eine Tiefensuche vorausgesetzt und davon ausgegangen, dass in jedem Verzweigungsschritt alle direkten Nachfolgerknoten auf einmal generiert werden. Der Beginn

des Verfahrens unterscheidet sich zu den bereits behandelten Branch-and-Bound-Algorithmen lediglich darin, dass der Stack Ω durch $\Omega := \{(\mathcal{C}, S, W, LB)\}$ initialisiert wird, wohingegen alle weiteren Operationen analog zu den bisherigen Verfahren ausgeführt werden. In jeder Iteration wird der oberste Knoten $(\mathcal{C}, S, W, LB)$ vom Stack Ω entnommen und durch $LB < UB$ überprüft, ob der Suchraum des Enumerationsknotens eine zulässige Lösung mit einer kürzeren Projektdauer als UB enthalten könnte. Falls die Bedingung $LB < UB$ gilt, werden Konsistenztests aus der Menge Γ^{D} oder Γ^{W} auf der Startzeitbeschränkung W ausgeführt. Gilt nach der Ausführung der Konsistenztests $\mathcal{S}_T^{UB}(W) = \emptyset$, wird der Knoten $(\mathcal{C}, S, W, LB)$ nicht weiter untersucht, da im Suchraum keine zulässige Lösung mit einer kürzeren Projektdauer als UB existiert. Andernfalls wird in Abhängigkeit von der Verzweigungsstrategie die Menge der einplanbaren Vorgänge $\mathcal{E}$ bestimmt und ein Vorgang $i \in \mathcal{E}$ für den Verzweigungsschritt gewählt. Im Anschluss daran werden analog zu Algorithmus 5.6 die Mengen Θ_i und T_i berechnet, wobei die Zeitpunkte in T_i in einer Reihenfolge gemäß der Generierungsstrategie durchlaufen werden. Jeder entnommene Zeitpunkt t aus der Menge T_i wird, wie bereits für das Enumerationsschema in Algorithmus 5.6 beschrieben, Vorgang i als frühestmöglichem Startzeitpunkt zugeordnet, sodass schließlich der Schedule S' und die Startzeitbeschränkung W' für den direkten Nachfolgerknoten initialisiert werden können. Es ist dabei zu beachten, dass die Operationen in Zeile 18 von Algorithmus 5.9 für die Anwendung der Techniken zur Redundanzvermeidung UPT und ULT entsprechend angepasst werden können (s. Abschnitt 5.3.3). Nach der Bestimmung des Schedules S' und der Startzeitbeschränkung W' werden die Konsistenztests aus der Menge Γ^{B} auf W' ausgeführt. Gilt nach der Ausführung der Konsistenztests $\mathcal{S}_T^{UB}(W') = \emptyset$, wird der direkte Nachfolgerknoten W' nicht weiter untersucht. Kann dagegen nicht ausgeschlossen werden, dass eine zulässige Lösung im Suchraum des Nachfolgerknotens mit einer kürzeren Projektdauer als UB existiert, wird überprüft, ob Vorgänge ausgeplant werden müssen oder Vorgang i eventuell eingeplant werden kann. Falls $\mathcal{C}' = V$ nach der Einplanung von Vorgang i gilt, stellt S' eine neue beste zulässige Lösung dar, sodass $UB := S'_{n+1}$ aktualisiert und $S^* := S'$ abgespeichert wird. Weiterhin werden aus der Menge T_i alle Zeitpunkte entfernt, die größer als t sind und durch die somit keine besseren zulässigen Lösungen generiert werden können. Gilt dagegen $\mathcal{C}' \neq V$, wird die untere Schranke LB' berechnet, die entweder der unteren Schranke $LB0^{\pi} = S'_{n+1}$ oder der destruktiven unteren Schranke LBD^{π} entspricht. Für die Bestimmung von LBD^{π} werden analog zum relaxationsbasierten Branch-and-Bound-Verfahren $LB^{start} := LB0^{\pi}$ und $UB^{start} := UB - 1$ gesetzt (s. Abschnitt 4.3). Im Fall $LB' < UB$ wird der direkte Nachfolgerknoten $(\mathcal{C}', S', W', LB')$ der Liste Λ hinzugefügt, die nach der Betrachtung aller Zeitpunkte in T_i dazu verwendet wird, um alle generierten direkten Nachfolgerknoten sortiert nach nichtsteigenden Werten von LB' auf den Stack

Ω zu legen. Das beschriebene Verfahren wird solange ausgeführt, bis alle Knoten im Enumerationsbaum untersucht wurden ($\Omega = \emptyset$). Nach Abschluss des konstruktionsbasierten Branch-and-Bound-Verfahrens gibt Algorithmus 5.9 entweder einen optimalen Schedule S^* zurück oder zeigt, falls $UB = \bar{d} + 1$ gilt, dass für die betrachtete Probleminstanz kein zulässiger Schedule existiert.

Kapitel 6

Zeitindexbasierte Modellformulierungen

Für das RCPSP wurden über die letzten Jahrzehnte verschiedene ganzzahlige und gemischt-ganzzahlige lineare Programme (ILPs und MILPs) entwickelt, die sich durch die Definition der verwendeten Entscheidungsvariablen und durch unterschiedliche Formulierungen der Zeit- und Ressourcenrestriktionen unterscheiden. Sowohl ein ILP als auch ein MILP kann allgemein durch eine lineare Zielfunktion und lineare Nebenbedingungen beschrieben werden, wobei ein ILP ausschließlich ganzzahlige und ein MILP sowohl ganzzahlige als auch reellwertige Entscheidungsvariablen enthält. Ein Überblick zu ILPs und MILPs, die in den letzten Jahrzehnten für das RCPSP entwickelt wurden, kann der Arbeit von Artigues et al. (2015) entnommen werden. Die darin beschriebenen linearen Programme lassen sich in Abhängigkeit der verwendeten Entscheidungsvariablen in zeitindexbasierte, reihenfolgebasierte und eventbasierte Formulierungen unterteilen.[4]

Im Folgenden werden drei zeitindexbasierte Modellformulierungen für das RCPSP/max-π vorgestellt, die aus bekannten ILPs zum RCPSP entwickelt werden. In den Modellen werden unterschiedliche zeitindexbasierte binäre Entscheidungsvariablen betrachtet, die dadurch gekennzeichnet sind, dass sie jedem Zeitpunkt des Planungshorizonts einen bestimmten Zustand zuordnen. In Artigues (2017) werden die zeitindexbasierten Modelle zum RCPSP, die im Folgenden betrachtet und für das RCPSP/max-π erweitert werden, im Hinblick auf ihre LP-Relaxationen miteinander verglichen und bewertet. Analog zu Artigues (2017) werden im weiteren Verlauf dieses Kapitels die zeitindexbasierten Entscheidungsvariablen und die zugehörigen Modelle sowohl für das RCPSP als auch für das RCPSP/max-π als Pulse-, Step- und On/Off-Variablen bzw. Formulierungen bezeichnet. An dieser Stelle sei noch erwähnt, dass die reihenfolge- und eventbasierten Modellformulierungen zum RCPSP für Anpassungen an das RCPSP/max-π allgemein nicht geeignet

[4] Artigues et al. (2015) unterteilen die Modellformulierungen zum RCPSP in die Kategorien „Time-indexed“, „Sequencing/natural-date“ und „Positional-date/assignment“.

zu sein scheinen, da sie spezifische Eigenschaften erneuerbarer Ressourcen ausnutzen, die durch partiell erneuerbare Ressourcen nicht erfüllt werden.

Im ersten ganzzahligen linearen Programm zum RCPSP/max-π werden Pulse-Variablen zur Modellierung der Zeit- und Ressourcenrestriktionen eingesetzt. Durch eine Pulse-Variable wird jedem ganzzahligen zeitzulässigen Startzeitpunkt $t \in \mathcal{T}_i$ eines Vorgangs $i \in V$ ein Zustand zugeordnet, der angibt, ob der jeweilige Vorgang i zum Zeitpunkt t startet oder nicht. Die entsprechende binäre Entscheidungsvariable x_{it} ist wie folgt definiert:

$$x_{it} := \begin{cases} 1, & \text{falls Vorgang } i \text{ zum Zeitpunkt } t \text{ startet} \\ 0, & \text{sonst} \end{cases}$$

Das erste zeitindexbasierte Modell zum RCPSP/max-π entspricht einer Erweiterung des ganzzahligen linearen Programms aus Böttcher et al. (1999) für das RCPSP/π, das wiederum auf der Formulierung aus Pritsker et al. (1969) für das RCPSP basiert. Die entsprechende Pulse-Formulierung lautet wie folgt:

$$\text{Minimiere} \quad \sum_{t \in \mathcal{T}_{n+1}} t\, x_{n+1,t} \tag{6.1}$$

$$\text{u. d. N.} \quad \sum_{t \in \mathcal{T}_i} x_{it} = 1 \qquad (i \in V) \tag{6.2}$$

$$\sum_{t \in \mathcal{T}_j} t\, x_{jt} - \sum_{t \in \mathcal{T}_i} t\, x_{it} \geq \delta_{ij} \qquad ((i,j) \in E) \tag{6.3}$$

$$\sum_{i \in V_k} r_{ik}^{d} \sum_{v \in \Pi_k} \sum_{t \in Q_{iv} \cap \mathcal{T}_i} x_{it} \leq R_k \qquad (k \in \mathcal{R}) \tag{6.4}$$

$$x_{it} \in \{0,1\} \qquad (i \in V,\ t \in \mathcal{T}_i) \tag{6.5}$$

Zunächst wird die zu minimierende Projektdauer S_{n+1} durch die Zielfunktion (6.1) angegeben, wobei zu beachten ist, dass der Zusammenhang $S_i = \sum_{t \in \mathcal{T}_i} t\, x_{it}$ für jeden Vorgang $i \in V$ gilt. Durch die Nebenbedingungen (6.2) wird sichergestellt, dass jeder Vorgang $i \in V$ genau zu einem zeitzulässigen Zeitpunkt startet und durch die Ungleichungen (6.3) werden die Zeitbeziehungen zwischen den Vorgängen des Projekts modelliert. Die linke Seite jeder der Ungleichungen (6.4) stellt die Inanspruchnahme einer Ressource $k \in \mathcal{R}$ durch alle Vorgänge des Projekts dar. Dabei ergibt sich aus der Definition der Entscheidungsvariablen zunächst der Zusammenhang

$$r_{ik}^{u}(S_i) = \sum_{v \in \Pi_k} \sum_{t \in Q_{iv} \cap \mathcal{T}_i} x_{it}$$

für jeden realen Vorgang $i \in V^r$ mit $Q_{iv} := [v - p_i, v - 1] \cap \mathbb{Z}$ als Menge der Startzeitpunkte, für die sich Vorgang i in der Periode $v \in \Pi_k$ in Ausführung befinden würde. Entsprechend stellen die Ungleichungen (6.4) die Kapazitätsrestriktionen dar. Durch die Nebenbedingungen (6.5) werden schließlich die Wertebereiche aller Entscheidungsvariablen festgelegt.

Durch die Modellierung der Zeitrestriktionen (6.3) in disaggregierter Form kann eine alternative Pulse-Formulierung bestimmt werden. Diese Art der Modellierung wurde erstmals in Christofides et al. (1987) für das RCPSP vorgestellt und kann für allgemeine Zeitbeziehungen durch

$$\sum_{\tau=ES_i}^{\min(t-\delta_{ij},LS_i)} x_{i\tau} - \sum_{\tau=ES_j}^{t} x_{j\tau} \geq 0 \qquad ((i,j) \in E,\ t \in \mathcal{T}_j) \tag{6.6}$$

angepasst werden. Indem die Nebenbedingungen (6.3) durch die disaggregierten Zeitrestriktionen (6.6) ersetzt werden, ergibt sich schließlich die disaggregierte Pulse-Formulierung für das RCPSP/max-π.

Im zweiten ganzzahligen linearen Programm für das RCPSP/max-π werden Step-Variablen verwendet, die jedem Vorgang $i \in V$ und jedem Zeitpunkt $t \in \mathcal{H}^+ := \mathcal{H} \cup \{-1, -2, \ldots, -\bar{d}\}$ des erweiterten Planungshorizonts einen Zustand zuordnen, der angibt, ob der Vorgang bereits gestartet wurde oder nicht. Die entsprechende binäre Entscheidungsvariable z_{it} wird durch

$$z_{it} := \begin{cases} 1, & \text{falls Vorgang } i \text{ zum Zeitpunkt } t \text{ oder früher startet} \\ 0, & \text{sonst} \end{cases}$$

definiert. Die Erweiterung des Planungshorizonts $\mathcal{H}^+$ um eine ausreichende Anzahl negativer Zeitpunkte wird zur Vereinfachung der Darstellung der Nebenbedingungen im nachfolgenden Programm betrachtet. Weiterhin wird der Zusammenhang $x_{it} = z_{it} - z_{i,t-1}$ für die Modellierung der Startzeitpunkte der Vorgänge ausgenutzt, wobei der Startzeitpunkt jedes Vorgangs $i \in V$ durch $S_i = \sum_{t \in \mathcal{T}_i} t\, \zeta_{it}$ mit $\zeta_{it} := z_{it} - z_{i,t-1}$ dargestellt werden kann. Für die ressourcenbeschränkte Projektplanung wurden ganzzahlige lineare Programme mit Step-Variablen bereits in Sankaran et al. (1999) und Klein (2000, Abschnitt 3.2.1) untersucht, wobei beide Programme Zeitrestriktionen in disaggregierter Form betrachten. Die Step-Formulierung für das RCPSP/max-π mit aggregierten Zeitrestriktionen, die bereits in Watermeyer und Zimmermann (2020) vorgestellt wurde, ist durch das folgende

mathematische Modell gegeben:

$$\text{Minimiere} \quad \sum_{t \in \mathcal{T}_{n+1}} t\, \zeta_{n+1,t} \tag{6.7}$$

$$\text{u. d. N.} \quad \sum_{t \in \mathcal{T}_j} t\, \zeta_{jt} - \sum_{t \in \mathcal{T}_i} t\, \zeta_{it} \geq \delta_{ij} \qquad ((i,j) \in E) \tag{6.8}$$

$$\sum_{i \in V_k} r_{ik}^d \sum_{t \in A_{ik}} (z_{it} - z_{i,t-p_i}) \leq R_k \qquad (k \in \mathcal{R}) \tag{6.9}$$

$$z_{i,t-1} \leq z_{it} \qquad (i \in V,\ t \in \mathcal{T}_i) \tag{6.10}$$

$$z_{it} = 0 \qquad (i \in V,\ t \in \mathcal{H}^+ : t < ES_i) \tag{6.11}$$

$$z_{it} = 1 \qquad (i \in V,\ t \in \mathcal{H}^+ : t \geq LS_i) \tag{6.12}$$

$$z_{it} \in \{0,1\} \qquad (i \in V,\ t \in \mathcal{H}^+) \tag{6.13}$$

Die Zielfunktion (6.7) und die Zeitrestriktionen (6.8) ergeben sich zunächst analog zur aggregierten Pulse-Formulierung aus dem Zusammenhang $x_{it} = \zeta_{it}$ bzw.

$$\sum_{t \in \mathcal{T}_i} t\, \zeta_{it} = \sum_{t \in \mathcal{T}_i} t\, x_{it}$$

für jeden Vorgang $i \in V$. Weiterhin stellen die Nebenbedingungen (6.9) die Einhaltung der Ressourcenrestriktionen sicher, wobei die Menge $A_{ik} := \{t \in \mathcal{T}_i^+ \mid t+1 \in \Pi_k\}$ mit $\mathcal{T}_i^+ := [ES_i, LS_i + p_i[\cap \mathbb{Z}$ für jeden realen Vorgang $i \in V^r$ alle möglichen Ausführungszeitpunkte $t \in \mathcal{T}_i^+$ umfasst, die zugleich einem Startzeitpunkt einer Periode $v \in \Pi_k$ entsprechen. Da $z_{it} - z_{i,t-p_i}$ für einen realen Vorgang $i \in V^r$ genau dann den Wert 1 annimmt, falls sich der Vorgang zum Zeitpunkt $t \in \mathcal{T}_i^+$ in Ausführung befindet, gilt der Zusammenhang

$$r_{ik}^u(S_i) = \sum_{t \in A_{ik}} (z_{it} - z_{i,t-p_i})$$

für jeden realen Vorgang $i \in V^r$ und jede Ressource $k \in \mathcal{R}$. Die linke Seite jeder der Ungleichungen (6.9) stellt somit die Inanspruchnahme einer Ressource $k \in \mathcal{R}$ durch alle Vorgänge des Projekts dar. Die Nebenbedingungen (6.10) und (6.12) stellen die Einhaltung der Definition der Step-Variablen sicher, wobei die Gleichungen (6.11) und (6.12) dafür sorgen, dass jeder Vorgang $i \in V$ nur zu einem zeitzulässigen Zeitpunkt $t \in \mathcal{T}_i$ starten kann. Abschließend werden durch die Nebenbedingungen (6.13) die Wertebereiche aller Entscheidungsvariablen festgelegt.

Wie bereits für die Pulse-Formulierung können die Zeitrestriktionen des ganzzahligen linearen Programms mit Step-Variablen auch in disaggregierter Form modelliert werden.

Dazu werden die Nebenbedingungen (6.8) durch die disaggregierten Zeitrestriktionen

$$z_{i,t-\delta_{ij}} - z_{jt} \geq 0 \qquad ((i,j) \in E,\ t \in \mathcal{H}) \tag{6.14}$$

ersetzt, wobei die Modellierung der disaggregierten Zeitbeziehungen an Klein (2000, Abschnitt 3.2.1) angelehnt ist. Die Step-Formulierung mit disaggregierten Zeitrestriktionen für das RCPSP/max-π kann entsprechend als Erweiterung des ganzzahligen linearen Programms für das RCPSP aus Klein (2000) angesehen werden.

Das letzte ganzzahlige lineare Programm, das in diesem Kapitel für das RCPSP/max-π vorgestellt wird, setzt für die Modellierung der Zeit- und Ressourcenrestriktionen On/Off-Variablen ein. Im Gegensatz zu den Pulse- und Step-Variablen, die sich ausschließlich auf die Startzeitpunkte der Vorgänge des Projekts beziehen, geben die On/Off-Variablen die Zeitpunkte an, zu denen sich entweder ein realer Vorgang in Ausführung befindet oder ein fiktiver Vorgang eintritt. Die On/Off-Variablen sind durch

$$y_{it} := \begin{cases} 1, & \text{falls sich Vorgang } i \text{ zum Zeitpunkt } t \text{ in Ausführung befindet} \\ & \text{oder zum Zeitpunkt } t \text{ eintritt} \\ 0, & \text{sonst} \end{cases}$$

für jeden Vorgang $i \in V$ und jeden Zeitpunkt $t \in \mathcal{T}_i' := [ES_i, LC_i] \cap \mathbb{Z}$ mit $LC_i := LS_i + p_i$ als spätestem zeitzulässigen Endzeitpunkt von Vorgang i definiert. Die erste On/Off-Formulierung wurde in Kaplan (1988) für das RCPSP mit unterbrechbaren Vorgängen vorgestellt und in Klein (2000, Abschnitt 3.2.1) für das RCPSP angepasst. In Anlehnung an das Modell in Klein (2000) kann die On/Off-Formulierung für das RCPSP/max-π wie folgt angegeben werden:

$$\text{Minimiere} \quad \sum_{t \in \mathcal{T}_{n+1}} t\, y_{n+1,t} \tag{6.15}$$

$$\text{u. d. N.} \quad \sum_{t \in \mathcal{T}_i} y_{it} = 1 \qquad (i \in V^e) \tag{6.16}$$

$$\sum_{t \in \mathcal{T}_i^+} y_{it} = p_i \qquad (i \in V^r) \tag{6.17}$$

$$(y_{it} - y_{i,t+1})\, p_i \leq \sum_{\tau=t-p_i+1}^{t} y_{i\tau} \qquad (i \in V^r,\ t \in F_i) \tag{6.18}$$

$$y_{jt}\, p_i' \leq \sum_{\tau=ES_i}^{t-\delta_{ij}+p_i'-1} y_{i\tau} \qquad ((i,j) \in E,\ t \in F_{ij}) \tag{6.19}$$

$$\sum_{i \in V_k} r_{ik}^d \sum_{t \in A_{ik}} y_{it} \leq R_k \qquad (k \in \mathcal{R}) \tag{6.20}$$

$$y_{i,LC_i} = 0 \qquad (i \in V^r) \tag{6.21}$$

$$y_{it} \in \{0,1\} \qquad (i \in V,\ t \in \mathcal{T}_i') \tag{6.22}$$

Die Zielfunktion (6.15) entspricht der Projektdauer, die durch das mathematische Programm minimiert wird. Durch die Nebenbedingungen (6.16) und (6.17) wird sichergestellt, dass jeder fiktive Vorgang des Projekts genau einmal eintritt, und dass sich jeder reale Vorgang für genau p_i Zeiteinheiten in Ausführung befindet. Die unterbrechungsfreie Ausführung jedes realen Vorgangs wird durch die Nebenbedingungen (6.18) gewährleistet. Dazu wird ausgenutzt, dass $y_{it} - y_{i,t+1}$ genau dann den Wert 1 annimmt, falls Vorgang i zum Zeitpunkt $t+1$ endet. Für jeden realen Vorgang $i \in V^r$ werden alle Zeitpunkte $t \in F_i := [EC_i - 1, LC_i[\cap \mathbb{Z}$ mit $EC_i := ES_i + p_i$ als frühestem zeitzulässigen Endzeitpunkt von Vorgang i betrachtet, zu denen Vorgang i spätestens noch aktiv sein kann. Um für jeden realen Vorgang $i \in V^r$ zu garantieren, dass $y_{it} - y_{i,t+1} = 1$ für mindestens einen Zeitpunkt $t \in F_i$ gilt, werden die Gleichungen (6.21) dem Modell hinzugefügt. Durch die Nebenbedingungen (6.19) werden die Zeitrestriktionen des Projekts sichergestellt. Um für jede Zeitbeziehung $(i,j) \in E$ sowohl fiktive als auch reale Vorgänge zu berücksichtigen, wird der Bezeichner $p_i' := \max(1, p_i)$ für jeden Vorgang $i \in V$ eingeführt. Zur Modellierung einer Zeitbeziehung $(i,j) \in E$ wird für jeden Zeitpunkt $t \in F_{ij} := [ES_j, LS_i + \delta_{ij}[\cap \mathbb{Z}$ eine Ungleichung aufgestellt. Die Nebenbedingung für jedes Vorgangspaar $(i,j) \in E$ und jeden Zeitpunkt $t \in F_{ij}$ stellt jeweils sicher, dass Vorgang i spätestens bis zum Zeitpunkt $t - \delta_{ij} + p_i'$ beendet wird (eintritt), falls Vorgang j zum Zeitpunkt t eintritt oder sich zum Zeitpunkt t noch in Ausführung befindet. Die Ressourcenrestriktionen, die durch die Nebenbedingungen (6.20) modelliert werden, ergeben sich direkt aus der Step-Formulierung unter Berücksichtigung des Zusammenhangs $y_{it} = z_{it} - z_{i,t-p_i}$ für jeden realen Vorgang $i \in V^r$, sodass analog

$$r_{ik}^u(S_i) = \sum_{t \in A_{ik}} y_{it}$$

folgt. Die Nebenbedingungen (6.22) legen abschließend die Wertebereiche aller Entscheidungsvariablen fest.

Kapitel 7

Experimentelle Performance-Analyse

Im Folgenden wird die Performance der in Kapitel 5 entwickelten Branch-and-Bound-Verfahren untersucht. Dazu werden in Abschnitt 7.1 zunächst die Testinstanzen vorgestellt, die für die experimentellen Untersuchungen eingesetzt wurden. In Abschnitt 7.2 werden die Branch-and-Bound-Verfahren einander gegenübergestellt, wobei zunächst auf die Performance für Testinstanzen zum RCPSP/max-π und auf die Auswirkung der verschiedenen Verbesserungstechniken der Lösungsverfahren eingegangen wird. Im Anschluss daran werden die Lösungsverfahren aus Kapitel 5 mit einem Branch-and-Bound-Algorithmus aus Böttcher et al. (1999) für das RCPSP/π anhand etablierter Testinstanzen verglichen. In Abschnitt 7.3 wird das Branch-and-Bound-Verfahren mit der besten Performance über alle Testsets dem MILP-Solver IBM CPLEX gegenübergestellt, der zur Lösung der zeitindexbasierten Modellformulierungen aus Kapitel 6 eingesetzt wird. Abschließend erfolgt in Abschnitt 7.4 ein Vergleich der Branch-and-Bound-Verfahren aus Kapitel 5 mit Näherungsverfahren, die für das RCPSP/π entwickelt wurden.

Alle in diesem Kapitel untersuchten Lösungsverfahren wurden auf einem PC mit einem Intel Core i7-8700 3,2 GHz Prozessor und 64 GB Arbeitsspeicher unter Windows 10 ausgeführt. Die Branch-and-Bound-Verfahren sowie die zeitindexbasierten Modellformulierungen wurden in C++ kodiert und mit dem 64-bit Visual Studio 2017 C++-Compiler kompiliert. Für die Kodierung der ganzzahligen linearen Programme wurde das IBM OPL C++-Interface verwendet und zur Lösung der zeitindexbasierten Modellformulierungen der MILP-Solver IBM CPLEX in der Version 12.8.0 unter Standardeinstellungen eingesetzt. Zur Gewährleistung eines fairen Vergleichs wurden sowohl die Branch-and-Bound-Verfahren als auch der MILP-Solver auf einem einzelnen Thread ausgeführt. Für die Untersuchungen wurde ein Zeitlimit von 300 Sekunden für alle Testinstanzen mit weniger als 100 realen Vorgängen und ein Zeitlimit von 600 Sekunden für alle größeren Testinstanzen festgelegt.

7.1 Testinstanzen

In diesem Abschnitt werden zunächst die Testinstanzen beschrieben, die zur Bewertung der Performance der unterschiedlichen Lösungsverfahren im weiteren Verlauf dieses Kapitels verwendet werden. Für den Vergleich der Lösungsverfahren werden sowohl Testinstanzen zum RCPSP/max-π als auch zum RCPSP/π betrachtet. Die Testinstanzen zum RCPSP/max-π, die im Folgenden herangezogen werden, wurden in Watermeyer und Zimmermann (2020) erzeugt und stellen die einzigen bislang verfügbaren Testinstanzen für das RCPSP/max-π dar. Die generierten Instanzen basieren auf den UBO-Testinstanzen, die in Franck et al. (2001b) für das RCPSP/max mit dem Instanzgenerator ProGen/max (s. Schwindt, 1996, 1998a) erzeugt wurden und über die PSPLIB (s. Kolisch und Sprecher, 1997) frei zur Verfügung stehen. Im Folgenden wird die Vorgehensweise zur Generierung der RCPSP/max-π-Instanzen erläutert, die auch in Watermeyer und Zimmermann (2020) beschrieben wird. Zunächst werden für jedes UBOn-Testset mit $n = 10, 20, 50, 100, 200$ realen Vorgängen jeweils drei Instanzen ausgewählt, die durch ProGen/max mit Werten von 0,25; 0,5 und 0,75 für die sogenannte Order Strength generiert wurden. Die Order Strength OS (s. Schwindt, 1998a, Definition 3) ist eine Näherung für die Restrictiveness eines Projektnetzplans nach Thesen (1977). OS kann als eine $[0, 1]$-normierte Kennzahl aufgefasst werden, die angibt, in welchem Maß die vorliegenden Zeitbeziehungen in einem Projektnetzplan die zeitzulässigen Ausführungsreihenfolgen der Vorgänge des Projekts einschränken. Dabei können für $OS = 0$ alle Vorgänge in Bezug auf die vorliegenden Zeitrestriktionen in beliebiger Reihenfolge gestartet werden, wohingegen für $OS = 1$ für jedes Vorgangspaar $(i, j) \in V \times V$ entweder $d_{ij} \geq 0$ oder $d_{ji} \geq 0$ gilt, sodass einer der Vorgänge nicht vor dem anderen starten kann. Da die Restrictiveness nicht effizient berechnet werden kann, wird stattdessen die Order Strength OS für ProGen/max verwendet, für die in Thesen (1977) in einer empirischen Studie gezeigt wurde, dass sie unter mehr als 40 untersuchten Näherungswerten zur Restrictiveness den kleinsten durchschnittlichen relativen Fehler aufweist. Es ist zu beachten, dass OS für den Instanzgenerator ProGen/max lediglich einen Zielwert vorgibt, der im Allgemeinen von der tatsächlichen Order Strength OS' des Projektnetzplans nach der Instanzgenerierung abweicht. Um bei der Auswahl der drei Testinstanzen aus jedem UBOn-Testset eine zu große Differenz zwischen OS' und OS zu vermeiden, wird jeweils die Instanz mit kleinster Nummerierung und einer maximalen Abweichung von 10 % vom Zielwert $OS = 0{,}25; 0{,}5; 0{,}75$ ausgewählt. Nach der Auswahl der UBO-Testinstanzen werden die enthaltenen erneuerbaren Ressourcen durch partiell erneuerbare Ressourcen ersetzt, wobei die Zeitbeziehungen zwischen den Vorgangspaaren des Projekts sowie die Ausführungsdauern der Vorgänge unverändert übernommen werden.

Für die Generierung der partiell erneuerbaren Ressourcen wird das Verfahren aus Schirmer (1999) angewendet, das zur Generierung von RCPSP/π-Instanzen entwickelt wurde und einer Erweiterung des Instanzgenerators ProGen von Kolisch et al. (1995) entspricht. Der in Schirmer (1999, Abschnitt 10.2) beschriebene Instanzgenerator ProGen/Π betrachtet drei $[0,1]$-normierte Kontrollparameter für die Bestimmung der Periodenmengen Π_k aller partiell erneuerbaren Ressourcen $k \in \mathcal{R}$, die als Horizon Factor *HF*, Cardinality Factor *CF* und Interval Factor *IF* bezeichnet werden. Der Horizon Factor *HF* bestimmt zunächst für alle Ressourcen $k \in \mathcal{R}$ eine obere Schranke $\bar{d}^{\mathcal{R}} := ES_{n+1}\,(1 - HF) + \bar{d} \cdot HF$ für die späteste Periode in Π_k. Abweichend vom Instanzgenerator ProGen/Π wird dabei zur Berücksichtigung der allgemeinen Zeitbeziehungen $\bar{d} := \sum_{i \in V} \max(p_i, \max_{(i,j) \in E} \delta_{ij})$ gesetzt.[5] In Abhängigkeit von $\bar{d}^{\mathcal{R}}$ wird durch den Cardinality Factor *CF* die Anzahl der Perioden in Π_k für jede Ressource $k \in \mathcal{R}$ durch $|\Pi| := 2\,(1 - CF) + (\bar{d}^{\mathcal{R}} - 1)\,CF$ festgelegt, woraus sich nach Lemma 10.2 in Schirmer (1999) eine obere Schranke $\bar{I} := \min(|\Pi|, \bar{d}^{\mathcal{R}} - |\Pi| + 1)$ für die Anzahl der Belegungsintervalle $\mathcal{I}_k$ für jede Ressource $k \in \mathcal{R}$ ergibt. Durch den Interval Factor *IF* wird für jede Ressource $k \in \mathcal{R}$ die Anzahl der Belegungsintervalle durch $\mathcal{I}_k := (1 - IF) + \bar{I} \cdot IF$ festgelegt. Zur Bestimmung der Periodenmenge Π_k für jede Ressource $k \in \mathcal{R}$ werden schließlich $|\Pi|$ Perioden auf die vorgegebenen $\mathcal{I}_k$ Belegungsintervalle verteilt und die Anzahl der Zeitperioden zwischen den Belegungsintervallen bestimmt.

Nach der Generierung der Periodenmengen Π_k aller Ressourcen $k \in \mathcal{R}$ werden im nächsten Schritt die Kapazitäten R_k und die Bedarfe r_{ik}^d aller Vorgänge $i \in V$ festgelegt. Zunächst wird der Resource Factor *RF* eingesetzt, um den durchschnittlichen Anteil aller Ressourcen, die durch einen realen Vorgang $i \in V^r$ in Anspruch genommen werden könnten ($r_{ik}^d > 0$), für die Instanzgenerierung vorzugeben. Der $[0,1]$-normierte Kontrollparameter *RF*, der durch

$$RF := \frac{1}{|V^r||\mathcal{R}|} \sum_{i \in V^r} \sum_{k \in \mathcal{R}} a_{ik} \qquad a_{ik} := \begin{cases} 1, & \text{falls } r_{ik}^d > 0 \\ 0, & \text{sonst} \end{cases}$$

definiert ist, stellt wie die Order Strength *OS* lediglich einen Zielwert für die Generierung einer Instanz dar, sodass der tatsächliche Resource Factor RF' nach der Instanzgenerierung von *RF* abweichen kann. Basierend auf einem vorgegebenen Zielwert *RF* wird analog zu Kolisch et al. (1995) jedem realen Vorgang $i \in V^r$ zunächst eine Anzahl $|\mathcal{R}_i| \in [\underline{a}, \overline{a}] \cap \mathbb{Z}$ an Ressourcen zugeordnet, für die $r_{ik}^d > 0$ gelten soll ($|\mathcal{R}_i| = \sum_{k \in \mathcal{R}} a_{ik}$). In einem nächsten Schritt wird dann der Ressourcenbedarf r_{ik}^d für jeden Vorgang i und

[5] Es ist zu beachten, dass in Schirmer (1999), abweichend von dem Vorgehen zur Generierung der Testinstanzen für das RCPSP/max-π, $\bar{d} := \bar{d}^{\mathcal{R}}$ nach der Bestimmung von $\bar{d}^{\mathcal{R}}$ gesetzt wird.

jede Ressource k mit $a_{ik} = 1$ einem Wert aus der Menge $[\underline{r}, \overline{r}] \cap \mathbb{Z}$ zugeordnet, wobei $r_{ik}^d := 0$ im Fall $a_{ik} = 0$ gesetzt wird. Die Werte für $|\mathcal{R}_i|$ und r_{ik}^d mit $a_{ik} = 1$ werden dabei basierend auf einer diskreten Gleichverteilung aus der jeweiligen Menge gewählt. Der letzte Kontrollparameter für die Instanzgenerierung ist die Resource Strength RS, die als $[0, 1]$-normierte Maßzahl für den Grad der Knappheit der Ressourcen des Projekts aufgefasst werden kann. In Abhängigkeit vom vorgegebenen Wert von RS werden die Kapazitäten aller Ressourcen $k \in \mathcal{R}$ durch

$$R_k := R_k^{\min} (1 - RS) + R_k^{\max} \cdot RS$$

mit $R_k^{\text{ext}} := \sum_{i \in V_k} \text{ext}\{r_{ik}^c(\tau) \mid ES_i \leq \tau \leq LS_i\}$ und $\text{ext} \in \{\min, \max\}$ bestimmt, sodass für $RS = 0$ die höchste und für $RS = 1$ die geringste Ressourcenknappheit vorliegt.

Für die Generierung der RCPSP/max-π-Instanzen wurden für jede UBO-Testinstanz, die wie beschrieben aus dem UBO-Testset entnommen wurde, die Werte 0,25; 0,5 und 0,75 für alle Kontrollparameter HF, CF, IF, RF und RS für die Erzeugung der verbleibenden Problemparameter verwendet. Für jede Instanz wurden genau 30 partiell erneuerbare Ressourcen erzeugt und die Werte $\underline{a} = 5$, $\overline{a} = 25$, $\underline{r} = 1$ und $\overline{r} = 10$ für die Bestimmung der Ressourcenbedarfe r_{ik}^d verwendet. Durch die beschriebene Vorgehensweise wurden jeweils 729 Testinstanzen mit $n = 10, 20, 50, 100, 200$ realen Vorgängen generiert. Im weiteren Verlauf dieser Arbeit wird mit UBO$n\pi$ das Testset bezeichnet, das alle 729 Instanzen mit $n = 10, 20, 50, 100, 200$ realen Vorgängen umfasst, wobei sich das UBOπ-Testset aus allen generierten Instanzen zusammensetzt.

Um die Performance der entwickelten Lösungsverfahren auch für das RCPSP/π bewerten zu können, werden in den nachfolgenden Abschnitten ebenfalls Testinstanzen für das RCPSP/π betrachtet, die von Böttcher et al. (1999) und Alvarez-Valdes et al. (2006) zur Bewertung exakter Verfahren sowie Heuristiken erzeugt wurden. Das erste dieser Testsets, das im weiteren Verlauf als Böttcher-Testset bezeichnet wird, setzt sich aus 2160 Instanzen mit 10 realen Vorgängen (P10π) und jeweils 250 Instanzen mit 15, 20, 25 und 30 realen Vorgängen (P15π, P20π, P25π, P30π) zusammen. Für jede Instanz des Böttcher-Testsets wurden jeweils 30 partiell erneuerbare Ressourcen erzeugt. Die Instanzgenerierung in Böttcher et al. (1999) unterscheidet sich von ProGen/Π fast ausschließlich nur dadurch, dass basierend auf ProGen für die Erzeugung des Projektnetzplans an Stelle von OS ein Zielwert für die durchschnittliche Anzahl nichtredundanter Zeitbeziehungen pro Vorgang vorgegeben wird. Das zweite Testset für das RCPSP/π wurde von Schirmer (1999) mit dem Instanzgenerator ProGen/Π generiert und später von Alvarez-Valdes et al. (2006) um größere Testinstanzen erweitert, die ebenfalls mit ProGen/Π erzeugt wurden. Die Testsets von Schirmer (1999) umfassen jeweils 960 Instanzen mit 10, 20, 30

und 40 realen Vorgängen (J10π, J20π, J30π, J40π), die alle mit 30 partiell erneuerbaren Ressourcen generiert wurden. Später wurde das Testset von Schirmer (1999) um 960 Instanzen mit 60 realen Vorgängen (J60π), die ebenfalls 30 partiell erneuerbare Ressourcen umfassen, von Alvarez-Valdes et al. (2006) erweitert. Alle Testinstanzen von Schirmer (1999) und Alvarez-Valdes et al. (2006) werden im Folgenden unter dem Begriff Schirmer-Alvarez-Valdes-Testset zusammengefasst. Es ist dabei zu beachten, dass 9 der 960 Instanzen des Testsets J10π, die in Schirmer (1999, Abschnitt 10.4) als unlösbar identifiziert werden konnten, in den nachfolgenden Untersuchungen nicht berücksichtigt werden, da sie nicht mehr verfügbar sind.

7.2 Untersuchung der Branch-and-Bound-Verfahren

In diesem Abschnitt werden die Branch-and-Bound-Verfahren aus Kapitel 5 zunächst mit Hilfe des UBOπ-Testsets miteinander verglichen. Im Anschluss daran werden die Lösungsverfahren anhand der Testinstanzen des Böttcher- und des Schirmer-Alvarez-Valdes-Testsets bewertet und dem einzigen bekannten Branch-and-Bound-Verfahren für das RCPSP/π gegenübergestellt.

Im Folgenden wird zunächst auf die Suchstrategien und Verbesserungstechniken der Branch-and-Bound-Verfahren eingegangen, die in Abhängigkeit der betrachteten Instanzgrößen für die Untersuchungen eingesetzt wurden. Die Spezifikation des Verfahrensablaufs des relaxationsbasierten Branch-and-Bound-Verfahrens (RBB) in Abhängigkeit des betrachteten Testsets UBO$n\pi$ mit $n = 10, 20, 50, 100, 200$ realen Vorgängen kann Tabelle 7.1 entnommen werden. Die Bezeichner, die in Tabelle 7.1 verwendet werden, stimmen größtenteils mit den Beschreibungen in den Kapiteln 4 und 5 überein, wobei lediglich einzelne zusätzliche Symbole verwendet werden, die im Folgenden erläutert werden. In der Zeile zur Traversierungsstrategie (Trav.-Strat.) wird für die Scattered-Path-Suche SPS$^+$ zunächst die vorgegebene Zeitspanne angegeben, nach der jeweils ein bislang noch nicht vollständig untersuchter Knoten mit kleinstem unteren Schrankenwert unter allen Knoten auf der kleinsten Ebene des Suchbaums als nächster Knoten betrachtet wird. Weiterhin geben die Werte in den eckigen Klammern die maximale Anzahl der erzeugten direkten Nachfolgerknoten in jedem Verzweigungsschritt für die Generierungsstrategie (Gener.-Strat.) sowie die Ebene im Suchbaum an, bis zu der die jeweiligen Konsistenztests ausgeführt werden (Konsistenztests). Falls keine Werte in eckigen Klammern angegeben werden, liegen dagegen keine Beschränkungen vor. Für die Reihenfolge- (Reihenf.-Strat.) und die Verzweigungsstrategie (Verzw.-Strat.) wird durch ext $\in \{\min, \max\}$

jeweils vorgegeben, ob kleinere (min) oder größere (max) Prioritätsregelwerte bzw. untere Schrankenwerte für die kürzeste Projektdauer vorgezogen werden. Abschließend wird in der letzten Zeile in Tabelle 7.1 angegeben, ob der Suchraum eines Enumerationsknotens in jedem Verzweigungsschritt, wie in Abschnitt 5.1.3 beschrieben, partitioniert wird („x“) oder die Enumeration wie in Abschnitt 5.1.1 vorgenommen wird („–“).

	UBO10π	UBO20π	UBO50π	UBO100π	UBO200π
Trav.-Strat.	DFS	SPS^{+} [2 s]	SPS^{+} [5 s]	SPS^{+} [15 s]	SPS^{+} [15 s]
Verzw.-Strat.	NCA(min)	NCA(min)	DST(max)	DST$^{\mathrm{I}}$(max)	DST$^{\mathrm{I}}$(max)
Gener.-Strat.	all	all	all	restrCL [10]	restrCL [10]
Reihenf.-Strat.	LB(min)	LB(min)	LB-DST(min)	LB-DST$^{\mathrm{I}}$(min)	LB-DST$^{\mathrm{I}}$(min)
Konsistenztests	$\gamma^{\infty}_{\mathrm{B}}$, $\gamma^{1}_{\mathrm{W}}[\mathcal{R}_i, 10]$	γ^{∞}_{B}, $\gamma^{1}_{W}[\mathcal{R}_i, 10]$	$\gamma^{\infty}_{\mathrm{B}}$, $\gamma^{1}_{\mathrm{D}}[\mathcal{R}_i]$	$\gamma^{\infty}_{\mathrm{B}}$, $\gamma^{1}_{\mathrm{D}}[\mathcal{R}_i, 5]$	γ^{∞}_{B}
Untere Schranke	LBD^{π}	LBD^{π}	LBD^{π}	$LB0^{\pi}$	$LB0^{\pi}$
Partitionierung	x	x	–	–	–

Tabelle 7.1: Spezifikation des Verfahrensablaufs von RBB in Abhängigkeit der Instanzgröße für das UBOπ-Testset
(Quelle: Watermeyer und Zimmermann (2020))

In Tabelle 7.2 sind die Spezifikationen des Verfahrensablaufs des konstruktionsbasierten Branch-and-Bound-Verfahrens (CBB), das in Abschnitt 5.3 eingeführt wurde, in Abhängigkeit der Instanzgröße für das UBOπ-Testset angegeben. Die Bezeichner in Tabelle 7.2, die in den Kapiteln 4 und 5 eingeführt wurden, werden durch die gleichen Symbole wie in Tabelle 7.1 ergänzt. Der einzige Unterschied zu Tabelle 7.1 besteht lediglich darin, dass in der letzten Zeile von Tabelle 7.2 Techniken angegeben werden, die zur Redundanzvermeidung (Red.-Vermeidung) im Enumerationsverlauf eingesetzt werden (s. Abschnitt 5.3.3). Da Voruntersuchungen zeigen konnten, dass die Bestimmung der eingeschränkten Menge der Einplanungszeitpunkte T_i durch Algorithmus 5.7 zu besseren Ergebnissen führt als durch Algorithmus 5.8, wird im weiteren Verlauf dieses Kapitels davon ausgegangen, dass T_i in jedem Verzweigungsschritt von CBB durch Algorithmus 5.7 ermittelt wird. Eine Gegenüberstellung der Performance für CBB über alle Testsets UBO$n\pi$, falls entweder Algorithmus 5.7 oder Algorithmus 5.8 zur Bestimmung von T_i verwendet wird, zeigt Tabelle A.1 im Anhang.

Für das partitionsbasierte Branch-and-Bound-Verfahren (PBB) wird für jede Testinstanz des UBOπ-Testsets eine Scattered-Path-Suche mit einer Zeitspanne von fünf Sekunden (SPS [5 s]) durchgeführt und in jedem Enumerationsknoten der Fixpunkt der Konsistenztests aus der Menge Γ^{T} bestimmt ($\gamma^{\infty}_{\mathrm{T}}$). Lediglich für die Verzweigungsstrategie werden unterschiedliche Verfahrensabläufe in Abhängigkeit von der Instanzgröße für PBB

	UBO10π	UBO20π	UBO50π	UBO100π	UBO200π
Trav.-Strat.	DFS	SPS [2 s]	SPS [5 s]	SPS [5 s]	SPS [15 s]
Verzw.-Strat.	$\bar{C}$, MRC(max)	$\bar{C}$, MRC(max)	$\prec_{\mathrm{D}}$, STd$^{\mathrm{I}}$(min)	$\prec_{\mathrm{C}}$, PF(max)	$\prec_{\mathrm{C}}$, PF(max)
Gener.-Strat.	restr-LT [10]	restr-LT [10]	restr-PV [5]	restr-PV [15]	restr-PV [30]
Reihenf.-Strat.	LB(min)	LB(min)	LB(min)	LB(min)	LB(min)
Konsistenztests	$\gamma_{\mathrm{B}}^{\infty}$	$\gamma_{\mathrm{B}}^{\infty}$, $\gamma_{\mathrm{D}}^{1}[\mathcal{R}_i, 2]$	$\gamma_{\mathrm{B}}^{\infty}$, $\gamma_{\mathrm{W}}^{1}[\mathcal{R}_i]$	$\gamma_{\mathrm{B}}^{\infty}$, $\gamma_{\mathrm{W}}^{1}[\mathcal{R}_i, 2]$	$\gamma_{\mathrm{B}}^{\infty}$
Untere Schranke	LBD^{π}	LBD^{π}	LBD^{π}	$LB0^{\pi}$	$LB0^{\pi}$
Red.-Vermeidung	UPT	UPT	UPT+ULT	UPT+ULT	UPT

Tabelle 7.2: Spezifikation des Verfahrensablaufs von CBB in Abhängigkeit der Instanzgröße für das UBOπ-Testset

betrachtet, indem für das Testset UBO10π die RCO-Regel und für alle weiteren Testinstanzen die RCR-Regel angewendet wird (s. Abschnitt 5.2.2).

Es ist zu beachten, dass die Spezifikationen der Verfahrensabläufe von RBB und CBB, die in den Tabellen 7.1 und 7.2 für das UBOπ-Testset angegeben werden, lediglich Ausführungsvarianten entsprechen, die in Voruntersuchungen einen guten Ausgleich zwischen der Anzahl optimal gelöster Instanzen und der Anzahl von Instanzen ohne ermittelten Lösungsstatus zeigen konnten. Entsprechend existieren i. d. R. sowohl für das RBB als auch das CBB Ausführungsvarianten, für die mehr Instanzen optimal gelöst werden können, wobei zugleich der Lösungsstatus von mehr Instanzen unbestimmt bleibt. Analog dazu gibt es für das RBB und das CBB i. d. R. ebenfalls Spezifikationen, durch die der Lösungsstatus von mehr Instanzen bestimmt werden kann, was allerdings mit einer Abnahme optimal gelöster Instanzen einhergeht. Dieser Trade-off, der charakteristisch für die meisten exakten Lösungsverfahren ist, konnte für das PBB dagegen nur zu einem geringen Maß in Voruntersuchungen beobachtet werden. Stattdessen wurde häufiger eine Erhöhung (Verringerung) der Anzahl optimal gelöster Instanzen bei einer zeitgleichen Zunahme (Abnahme) der Anzahl der Instanzen mit ermitteltem Lösungsstatus festgestellt. Ein weiterer Vorteil von PBB gegenüber RBB und CBB, der sich in Voruntersuchungen herausgestellt hat, ergibt sich daraus, dass eine Spezifikation des Verfahrensablaufs von PBB, die gegenüber einer anderen Spezifikation für ein UBO$n\pi$-Testset vorteilhaft ist, überwiegend auch bessere Ergebnisse für alle anderen Testsets liefert. Im Gegensatz dazu sind die Spezifikationen der Verfahrensabläufe von RBB und CBB, wie aus den Tabellen 7.1 und 7.2 hervorgeht, an die Instanzgrößen anzupassen, worauf im Folgenden noch näher eingegangen wird.

Zunächst kann Tabelle 7.1 entnommen werden, dass die Partitionierung des Suchraums eines Enumerationsknotens in jedem Verzweigungsschritt lediglich für kleinere Testin-

stanzen (UBO10π, UBO20π) vorteilhaft ist. Dagegen konnten Voruntersuchungen zeigen, dass die Partitionierung bei größeren Testinstanzen zu einer starken Zunahme der Anzahl von Instanzen ohne bestimmten Lösungsstatus führen. Der Grund dafür könnte sein, dass aufgrund der redundanzfreien Enumeration, die durch die Partitionierung erzeugt wird, die Wahrscheinlichkeit abnimmt, dass in bestimmten Teilen des Suchbaums zulässige Lösungen vorliegen. Weiterhin geht aus Tabelle 7.1 hervor, dass mit der Zunahme der Instanzgröße die Intensität des Einsatzes von Konsistenztests sowie die eingesetzte Berechnungszeit für die unteren Schrankenwerte in jedem Enumerationsknoten verringert werden sollten. Es ist außerdem zu beachten, dass die Scattered-Path-Suche bereits für Instanzen mit mehr als zehn realen Vorgängen vorteilhaft ist, und dass die besten Prioritätsregeln für die Verzweigungs- und Reihenfolgestrategie von den Instanzgrößen abhängen. Insbesondere konnten Voruntersuchungen zeigen, dass ressourcenbasierte Prioritätsregeln (z. B. ACO, RCO und NCA) für kleinere Instanzen und zeitplanungsbasierte Prioritätsregeln (z. B. DST, TF und EFF) für größere Testinstanzen vorteilhaft sind.

Aus Tabelle 7.2 geht zunächst wie aus Tabelle 7.1 hervor, dass die Scattered-Path-Suche bereits für Instanzen mit mehr als zehn realen Vorgängen angewendet und die Berechnung der destruktiven unteren Schranke LBD^{π} auf kleinere Testinstanzen beschränkt werden sollte. Weiterhin konnten Voruntersuchungen analog zu RBB auch für das CBB zeigen, dass ressourcenbasierte Prioritätsregeln (z. B. MRC, TMAR und AMAR) für kleinere Testinstanzen und zeitplanungsbasierte Prioritätsregeln (z. B. LS und ST) für größere Testinstanzen vorteilhaft sind, wobei darüber hinaus netzplanbasierte Prioritätsregeln (z. B. PF und TS) die besten Ergebnisse für die größten Testinstanzen zeigen konnten. Für die Verzweigungsstrategie ist weiterhin die Verwendung der Distanz- oder Zyklenordnung ($\prec_{\mathrm{D}}, \prec_{\mathrm{C}}$) nur für größere Testinstanzen vorteilhaft, wohingegen bessere Ergebnisse für kleinere Testinstanzen erzielt werden, wenn alle aktuell nicht eingeplanten Vorgänge ($\bar{\mathcal{C}}$) betrachtet werden. Es ist außerdem zu beachten, dass für größere Testinstanzen Prioritätsregeln (PV) für die Bestimmung der Reihenfolge, in der die Zeitpunkte in T_i zur Generierung aller direkten Nachfolgerknoten durchlaufen werden, verwendet werden sollten, und dass die Redundanzvermeidungstechnik UPT über alle Instanzgrößen zu besseren Ergebnissen führt.

Abschließend sei noch erwähnt, dass insbesondere der RB-Konsistenztest für die Performance aller Branch-and-Bound-Verfahren eine entscheidende Rolle spielt. Aus den Tabellen 7.1 und 7.2 geht zunächst hervor, dass die besten Ergebnisse für das RBB und das CBB, unabhängig von der Größe der betrachteten Testinstanzen, durch die Bestimmung des Fixpunkts des RB- und TB-Konsistenztests ($\gamma_{\mathrm{B}}^{\infty}$) in jedem Enumerationsknoten erzielt werden. Für das PBB werden dagegen die besten Ergebnisse dadurch erreicht, dass

der RB-Konsistenztest mit dem T-Konsistenztest kombiniert wird bzw. der Fixpunkt der Konsistenztests der Menge Γ^{T} in jedem Enumerationsknoten berechnet wird ($\gamma_{\mathrm{T}}^{\infty}$). Durch die zusätzliche Betrachtung der D- und W-Konsistenztests bzw. durch die Ausführung der Konsistenztests in den Mengen Γ^{D} und Γ^{W} können dagegen nur leichte Performance-Verbesserungen für einige Testsets für das RBB und das CBB erzielt werden.

Tabelle 7.3 zeigt die Performance aller Branch-and-Bound-Verfahren für das UBOπ-Testset. In der dritten Spalte wird zunächst für jedes Testset UBO$n\pi$ die Anzahl aller enthaltenen nicht-trivialen Instanzen angegeben (#nTriv), wobei in Anlehnung an Alvarez-Valdes et al. (2008) eine Instanz trivial genannt wird, falls der *ES*-Schedule zulässig ist und somit bereits eine optimale Lösung darstellt. Da jede triviale Instanz bereits im Wurzelknoten in jedem der betrachteten Branch-and-Bound-Verfahren effizient gelöst wird, beschränken sich alle Untersuchungen im weiteren Verlauf dieses Kapitels ausschließlich auf alle nicht-trivialen Instanzen der Testsets. Die nächste Spalte in Tabelle 7.3 gibt an, für wie viele der nicht-trivialen Testinstanzen eine optimale Lösung innerhalb des vorgegebenen Zeitlimits durch das jeweilige Branch-and-Bound-Verfahren bestimmt und zugleich auch als optimal verifiziert wurde (#opt). Neben den optimal gelösten Instanzen wird weiterhin die Anzahl der Testinstanzen angegeben, für die entweder eine zulässige Lösung gefunden wurde (#zul) oder für die gezeigt werden konnte, dass keine zulässige Lösung existiert (#unl). Der letzte Anteil der nicht-trivialen Instanzen ist schließlich durch die Anzahl der Testinstanzen gegeben, deren Lösungsstatus unbestimmt bleibt, d. h., für die keine zulässige Lösung bestimmt wurde, aber auch nicht gezeigt werden konnte, dass keine zulässige Lösung existiert (#unb). Die letzten beiden Spalten in der Tabelle geben schließlich die durchschnittlichen Laufzeiten über alle Instanzen an, die durch das jeweilige Branch-and-Bound-Verfahren entweder optimal gelöst wurden ($\varnothing_{\mathrm{opt}}^{\mathrm{cpu}}$) oder die als unlösbar verifiziert werden konnten ($\varnothing_{\mathrm{unl}}^{\mathrm{cpu}}$).

Aus den Ergebnissen in Tabelle 7.3 geht zunächst hervor, dass alle Branch-and-Bound-Verfahren in der Lage sind, das Testset UBO10π vollständig zu lösen, wobei PBB die kürzesten Laufzeiten aufweist. Weiterhin zeigt PBB die besten Ergebnisse über alle Testsets UBO$n\pi$, sodass PBB das beste Verfahren zur Lösung der Testinstanzen des UBOπ-Testsets darstellt. Der Vergleich zwischen CBB und RBB zeigt darüber hinaus, dass der konstruktionsbasierte Ansatz für alle Testsets bis zu 100 realen Vorgängen besser geeignet ist als der relaxationsbasierte Ansatz von RBB. Dabei ist zu beachten, dass dieser Vorteil insbesondere für alle Testsets mit Instanzen mit bis zu 50 realen Vorgängen zu beobachten ist, wohingegen bereits für Testinstanzen mit 100 realen Vorgängen nur noch leicht bessere Ergebnisse von CBB erzielt werden können. Für das Testset UBO200π liegt schließlich keine eindeutige Dominanz mehr von CBB gegenüber RBB vor. Wäh-

		#nTriv	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$
	PBB		534	534	159	0	0,018 s	0,004 s
UBO10π	CBB	693	534	534	159	0	0,067 s	0,056 s
	RBB		534	534	159	0	0,040 s	0,004 s
	PBB		542	581	40	0	7,572 s	0,583 s
UBO20π	CBB	621	537	581	40	0	7,846 s	0,702 s
	RBB		500	578	40	3	8,076 s	8,006 s
	PBB		238	496	6	25	13,938 s	25,092 s
UBO50π	CBB	527	183	491	5	31	13,774 s	26,827 s
	RBB		145	486	3	38	8,022 s	0,279 s
	PBB		136	474	0	10	25,426 s	–
UBO100π	CBB	484	88	472	0	12	28,693 s	–
	RBB		82	467	0	17	39,700 s	–
	PBB		124	466	0	0	33,223 s	–
UBO200π	CBB	466	96	458	0	8	36,908 s	–
	RBB		82	466	0	0	43,292 s	–

Tabelle 7.3: Performance der Branch-and-Bound-Verfahren für das UBOπ-Testset

rend CBB im Vergleich zu RBB mehr Testinstanzen mit 200 realen Vorgängen optimal lösen kann, bestimmt CBB dagegen für acht der Testinstanzen, die von RBB (und PBB) als lösbar identifiziert werden konnten, keine zulässige Lösung. Es ist zu beachten, dass dieser Nachteil von CBB nahelegt, dass für die Entwicklung von Näherungsverfahren für das RCPSP/max-π neben konstruktionsbasierten Ansätzen auch Verfahren für die Generierung von Schedules untersucht werden sollten, die auf den relaxationsbasierten Enumerationsansätzen von PBB und RBB basieren. Weiterhin zeigen die Ergebnisse in Tabelle 7.3, dass die Schärfe der Ressourcenbeschränkungen mit steigender Anzahl an Vorgängen abfällt. Dieser Effekt konnte bereits bei anderen Problemstellungen der Projektplanung festgestellt werden und sollte bei der zukünftigen Generierung von Testinstanzen stärker berücksichtigt werden.

Zur näheren Untersuchung der Fähigkeit der unterschiedlichen Branch-and-Bound-Verfahren, eine lösbare Instanz zu identifizieren bzw. eine zulässige Lösung zu bestimmen, wird in Tabelle 7.4 zunächst für jedes Testset UBO$n\pi$ die Anzahl an Instanzen angegeben, für die entweder mindestens ein Verfahren ($\#^{\cup}_{\text{zul}}$) oder alle Verfahren ($\#^{\cap}_{\text{zul}}$) eine zulässige Lösung bestimmen konnten. Die weiteren Angaben in Tabelle 7.4 entsprechen der Anzahl an Testinstanzen, die durch mindestens ein Branch-and-Bound-Verfahren als lösbar identifiziert wurden, aber für die keine zulässige Lösung durch PBB ($\#^{\cup,\text{PBB}}_{\text{unb}}$), CBB ($\#^{\cup,\text{CBB}}_{\text{unb}}$) oder RBB ($\#^{\cup,\text{RBB}}_{\text{unb}}$) ermittelt werden konnte.

Die Ergebnisse in Tabelle 7.4 zeigen, dass PBB für alle Testinstanzen eine zulässige Lösung bestimmt, die auch durch CBB oder RBB als lösbar identifiziert werden konnten.

Testset	$\#^{\cup}_{\text{zul}}$	$\#^{\cap}_{\text{zul}}$	$\#^{\cup,\text{PBB}}_{\text{unb}}$	$\#^{\cup,\text{CBB}}_{\text{unb}}$	$\#^{\cup,\text{RBB}}_{\text{unb}}$
UBO20π	581	578	0	0	3
UBO50π	496	486	0	5	10
UBO100π	474	467	0	2	7
UBO200π	466	458	0	8	0

Tabelle 7.4: Vergleich zur Bestimmung lösbarer Instanzen für das UBOπ-Testset

Entsprechend werden CBB und RBB auch im Hinblick auf die Fähigkeit, lösbare Instanzen zu identifizieren bzw. zulässige Lösungen zu bestimmen, durch PBB dominiert. Weiterhin kann Tabelle 7.4 entnommen werden, dass diese Art der Dominanz auch für CBB im Vergleich zu RBB für alle Testsets mit bis zu 100 realen Vorgängen gilt, wohingegen CBB durch RBB für das Testset UBO200π dominiert wird.

Als Nächstes werden die Zielfunktionswerte der besten zulässigen Lösungen, die durch PBB, CBB und RBB ermittelt wurden, einander gegenübergestellt. Dazu werden in Tabelle 7.5 die Projektdauern der besten zulässigen Lösungen, die innerhalb des vorgegebenen Zeitlimits von PPB, CBB und RBB bestimmt wurden, miteinander verglichen. Die zweite Spalte gibt zunächst für jedes Testset UBO$n\pi$ die Anzahl der Testinstanzen an, für die durch alle Lösungsverfahren eine zulässige Lösung ermittelt wurde, aber für die nicht alle Verfahren eine optimale Lösung bestimmen und verifizieren konnten ($\#^{\cap,\text{nv}}_{\text{zul}}$). Für diese Instanzen wird in den darauffolgenden Spalten jeweils der Anteil angegeben, für den PBB ($p^{\#,\text{PBB}}_{\text{best}}$), CBB ($p^{\#,\text{CBB}}_{\text{best}}$) oder RBB ($p^{\#,\text{RBB}}_{\text{best}}$) eine beste zulässige Lösung, d. h. eine Lösung mit kürzester Projektdauer über alle Lösungsverfahren bestimmen konnte. Die durchschnittliche relative Abweichung über alle betrachteten Testinstanzen ($\#^{\cap,\text{nv}}_{\text{zul}}$) der Projektdauer der besten ermittelten zulässigen Lösung von PBB, CBB oder RBB von der Projektdauer einer besten Lösung über alle Lösungsverfahren kann den letzten Spalten der Tabelle 7.5 entnommen werden ($\varnothing^{\Delta,\text{PBB}}_{\text{best}}$, $\varnothing^{\Delta,\text{CBB}}_{\text{best}}$, $\varnothing^{\Delta,\text{RBB}}_{\text{best}}$).

Testset	$\#^{\cap,\text{nv}}_{\text{zul}}$	$p^{\#,\text{PBB}}_{\text{best}}$	$p^{\#,\text{CBB}}_{\text{best}}$	$p^{\#,\text{RBB}}_{\text{best}}$	$\varnothing^{\Delta,\text{PBB}}_{\text{best}}$	$\varnothing^{\Delta,\text{CBB}}_{\text{best}}$	$\varnothing^{\Delta,\text{RBB}}_{\text{best}}$
UBO20π	79	97,47 %	58,23 %	32,91 %	0,04 %	3,62 %	5,70 %
UBO50π	344	94,77 %	26,45 %	6,69 %	0,52 %	4,88 %	12,50 %
UBO100π	390	97,44 %	7,69 %	3,08 %	0,25 %	10,44 %	15,62 %
UBO200π	379	98,94 %	5,54 %	1,32 %	0,03 %	18,16 %	18,65 %

Tabelle 7.5: Vergleich der Lösungsgüten für das UBOπ-Testset

Die Ergebnisse in Tabelle 7.5 zeigen zunächst, dass PBB im Vergleich zu CBB und RBB über alle Testsets UBO$n\pi$ für den größten Anteil aller betrachteten Testinstanzen ($\#^{\cap,\text{nv}}_{\text{zul}}$) die besten Lösungen bestimmt, wobei die Differenz zu den Anteilen von CBB und RBB mit steigender Instanzgröße zunimmt. Weiterhin geht aus den letzten drei Spalten der Tabelle hervor, dass die durchschnittliche Lösungsgüte von PBB über alle

Testsets UBO$n\pi$ besser als die der anderen Lösungsverfahren ist, wobei die Differenz zu CBB und RBB auch für diesen Vergleichswert mit steigender Instanzgröße zunimmt. Im Hinblick auf die Lösungsgüte kann weiterhin eine Dominanz von CBB gegenüber RBB über alle Testsets UBO$n\pi$ festgestellt werden, wobei für das Testset UBO200π zu beachten ist, dass CBB für acht der Testinstanzen keine zulässige Lösung bestimmen konnte (s. Tabelle 7.4).

Abschließend soll für das UBOπ-Testset untersucht werden, welchen Einfluss die unterschiedlichen Verbesserungstechniken auf die Performance der Branch-and-Bound-Verfahren haben. Dazu werden im Folgenden die Testsets UBO20π und UBO50π betrachtet. Zunächst wird für jedes Branch-and-Bound-Verfahren eine Basisvariante (BV) ausgeführt, die den Suchbaum nach der zu Beginn dieses Abschnitts beschriebenen Suchstrategie aufbaut und lediglich den unteren Schrankenwert $LB0^{\pi}$ in jedem Enumerationsknoten bestimmt. Um den Einfluss der Verbesserungstechniken zu untersuchen, werden den Basisvarianten schrittweise die einzelnen Verbesserungstechniken der jeweiligen Lösungsverfahren hinzugefügt. Dabei ist zu beachten, dass für die jeweiligen Branch-and-Bound-Verfahren lediglich die Verbesserungstechniken untersucht werden, die auch in den bisherigen experimentellen Analysen für die Testsets UBO20π und UBO50π eingesetzt wurden (s. Tabellen 7.1 und 7.2 für RBB und CBB).

Die Tabellen 7.6 und 7.7 zeigen den Einfluss der Verbesserungstechniken auf die Performance von RBB. Neben den Vergleichswerten aus Tabelle 7.3 wird für jede Ausführungsvariante von RBB die gesamte Berechnungszeit t_{cpu} über alle Instanzen des jeweils betrachteten Testsets angegeben. Für das Testset UBO20π werden neben den Verbesserungstechniken, die in Tabelle 7.1 aufgeführt sind, auch die Auswirkungen der Dominanzregeln aus Abschnitt 5.1.2 untersucht, die als Alternativen zur Partitionierung ausgeführt werden.

	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$	t_{cpu}
RBB (BV)	191	562	7	52	9,502 s	8,552 s	128.775 s
+Preprocessing	268	566	21	34	8,141 s	0,011 s	101.782 s
$+LBD^{\pi}$	290	572	21	28	5,352 s	0,011 s	94.552 s
+Konsistenztests	373	579	24	18	3,566 s	0,377 s	68.539 s
$+\bar{U}$-Dominanz	394	579	27	15	4,218 s	1,843 s	61.712 s
$+W$-Dominanz	403	573	27	21	6,596 s	1,844 s	60.008 s
+Partitionierung	500	578	40	3	8,076 s	8,006 s	28.658 s

Tabelle 7.6: Untersuchung des Einflusses der Verbesserungstechniken auf die Performance von RBB für das Testset UBO20π

	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$	t_{cpu}
RBB (BV)	97	475	0	52	9,723 s	–	129.943 s
+Preprocessing	109	477	3	47	9,578 s	0,279 s	125.545 s
$+LBD^{\pi}$	116	486	3	38	9,065 s	0,279 s	123.452 s
+Konsistenztests	145	486	3	38	8,022 s	0,279 s	114.864 s

Tabelle 7.7: Untersuchung des Einflusses der Verbesserungstechniken auf die Performance von RBB für das Testset UBO50π

Die Ergebnisse in den Tabellen 7.6 und 7.7 zeigen zunächst, dass sowohl durch das Preprocessing als auch durch die Konsistenztests die Performance von RBB deutlich verbessert wird, wobei die Konsistenztests einen größeren Einfluss auf die Leistungsfähigkeit des Verfahrens haben. Die Berechnung der destruktiven unteren Schranke LBD^{π} führt dagegen nur zu leichten Performance-Verbesserungen für beide Testsets. Weiterhin geht aus Tabelle 7.6 für das Testset UBO20π hervor, dass auch die Dominanzregeln die Performance von RBB verbessern können, wobei durch die Partitionierung deutlich bessere Ergebnisse als durch den gemeinsamen Einsatz der $\bar{U}$- und W-Dominanzregel erzielt werden. Es ist dabei insbesondere zu beachten, dass erst durch die Partitionierung alle unlösbaren Instanzen des Testsets UBO20π bestimmt werden können.

Um für CBB zusätzlich den Einfluss der Berechnung der Einschränkung T_i auf die Performance zu untersuchen, werden zwei unterschiedliche Basisvarianten für CBB angenommen, die alle Zeitpunkte der Menge Θ_i oder T_i in einem Verzweigungsschritt zur Generierung der direkten Nachfolgerknoten betrachten.

	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$	t_{cpu}
CBB (BV Θ_i)	120	546	3	72	29,134 s	49,019 s	153.043 s
CBB (BV T_i)	142	560	5	56	25,654 s	65,402 s	146.170 s
+Preprocessing	214	567	22	32	23,165 s	0,493 s	120.468 s
$+LBD^{\pi}$	222	567	22	32	24,408 s	1,010 s	118.541 s
+Konsistenztests	343	576	23	22	10,667 s	0,363 s	80.167 s
+UPT	537	581	40	0	7,846 s	0,702 s	17.442 s

Tabelle 7.8: Untersuchung des Einflusses der Verbesserungstechniken auf die Performance von CBB für das Testset UBO20π

Zunächst geht aus den Tabellen 7.8 und 7.9 hervor, dass durch die Berechnung der Einschränkung T_i insbesondere für das Testset UBO20π eine Performance-Verbesserung erreicht wird, wohingegen für das Testset UBO50π lediglich eine Verbesserung der Berechnungszeiten erzielt werden kann. Analog zu RBB können durch das Preprocessing und die Konsistenztests deutliche Verbesserungen erreicht werden. Die Berechnung der destruktiven unteren Schranke LBD^{π} kann dagegen nur die Performance für das Testset

	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$	t_{cpu}
CBB (BV Θ_i)	72	489	0	38	7,822 s	–	137.063 s
CBB (BV T_i)	72	489	0	38	4,961 s	–	136.857 s
+Preprocessing	81	486	3	38	9,934 s	0,383 s	133.706 s
+LBD^{π}	80	486	3	38	9,903 s	0,383 s	133.993 s
+Konsistenztests	120	491	3	33	22,439 s	0,383 s	123.894 s
+UPT	183	491	5	31	17,162 s	28,858 s	104.985 s
+ULT	183	491	5	31	13,774 s	26,827 s	104.355 s

Tabelle 7.9: Untersuchung des Einflusses der Verbesserungstechniken auf die Performance von CBB für das Testset UBO50π

UBO20π verbessern, wohingegen LBD^{π} sogar einen negativen Einfluss auf die Performance von CBB für das Testset UBO50π hat. Im Gegensatz zu RBB führt die Redundanzvermeidung sowohl für das Testset UBO20π als auch für UBO50π zu deutlichen Performance-Verbesserungen, wobei die zusätzliche Ausführung von ULT für das Testset UBO50π und die daraus resultierende redundanzfreie Enumeration lediglich die Berechnungszeiten verringern kann.

Die Ergebnisse in den Tabellen 7.10 und 7.11 zeigen den Einfluss des Preprocessings und der Konsistenztests auf die Performance von PBB für die Testsets UBO20π und UBO50π.

	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$	t_{cpu}
PBB (BV)	525	579	40	2	7,284 s	2,155 s	20.710 s
+Preprocessing	526	579	40	2	7,432 s	2,262 s	20.499 s
+Konsistenztests	542	581	40	0	7,572 s	0,583 s	15.827 s

Tabelle 7.10: Untersuchung des Einflusses der Verbesserungstechniken auf die Performance von PBB für das Testset UBO20π

	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$	t_{cpu}
PBB (BV)	221	494	5	28	15,608 s	5,720 s	93.778 s
+Preprocessing	219	494	5	28	13,923 s	14,221 s	94.020 s
+Konsistenztests	238	496	6	25	13,938 s	25,092 s	88.368 s

Tabelle 7.11: Untersuchung des Einflusses der Verbesserungstechniken auf die Performance von PBB für das Testset UBO50π

Zunächst kann festgestellt werden, dass die Basisvariante von PBB bereits eine gute Performance im Vergleich zu RBB und CBB aufweist. Die Basisvariante von PBB kann dabei für das Testset UBO50π sogar bessere Ergebnisse als die besten Ausführungsvarianten von RBB und CBB erzielen. Die weiteren Ergebnisse in den Tabellen 7.10 und 7.11 zeigen weiterhin, dass das Preprocessing nur einen geringen Einfluss auf die Performance von

PBB hat, wobei sogar ein negativer Einfluss für das Testset UBO50π festgestellt werden kann. Die weiteren Konsistenztests führen dagegen zu deutlichen Performance-Verbesserungen, wobei die Konsistenztests allerdings einen geringeren Stellenwert für PBB als für RBB und CBB haben.

Im letzten Teil dieses Abschnitts werden die Branch-and-Bound-Verfahren PBB, CBB und RBB mit dem einzigen bekannten Branch-and-Bound-Algorithmus für das RCPSP/π (BOT) von Böttcher (1995) bzw. von Böttcher et al. (1999) verglichen. Das Lösungsverfahren BOT beruht auf dem konstruktionsbasierten Enumerationsansatz des Branch-and-Bound-Verfahrens von Talbot und Patterson (1978), dem zwei Zulässigkeitsschranken FB1 und FB2 (engl. Feasibility Bound) hinzugefügt werden (vgl. Böttcher et al., 1999). Durch die Zulässigkeitsschranken FB1 und FB2 werden Teilschedules vorzeitig von den weiteren Untersuchungen im Enumerationsverlauf ausgeschlossen, die entweder zu keiner zulässigen oder zu keiner besseren als einer bereits bekannten zulässigen Lösung erweitert werden können. Da der Originalcode von BOT nicht zur Verfügung gestellt werden konnte, wurde BOT nach den Beschreibungen in Böttcher (1995, Kapitel 4 und 5) und Böttcher et al. (1999) nachimplementiert. Für die nachfolgenden Untersuchungen wird BOT mit beiden Zulässigkeitsschranken FB1 und FB2 ausgeführt. Es ist dabei zu beachten, dass Voruntersuchungen zeigen konnten, dass durch die gemeinsame Ausführung von FB1 und FB2 die besten Ergebnisse für BOT über alle Testsets für das RCPSP/π erzielt werden. Ein Vergleich der verschiedenen Ausführungsvarianten für BOT in Bezug auf die Zulässigkeitsschranken kann den Tabellen A.2 und A.3 im Anhang entnommen werden.

Die Tabellen 7.12 und 7.13 zeigen die Performance aller Branch-and-Bound-Verfahren für das Böttcher- und das Schirmer-Alvarez-Valdes-Testset. Für die Untersuchungen wurden RBB und CBB mit den gleichen Spezifikationen wie für das Testset UBO20π ausgeführt, mit der einzigen Ausnahme, dass CBB für die Testsets P25π und J60π mit dem Verfahrensablauf für das Testset UBO50π durchgeführt wurde (s. Tabellen 7.1 und 7.2). Für das PBB wurde in jedem Enumerationsknoten der Fixpunkt der Konsistenztests aus der Menge Γ^{B} bestimmt ($\gamma_{\mathrm{B}}^{\infty}$), die RCO-Regel für die Testsets P10π, P15π, P20π und J10π sowie die RCR-Regel für alle weiteren Testsets angewendet.

Zunächst geht aus den Ergebnissen in den Tabellen 7.12 und 7.13 hervor, dass BOT von allen anderen Branch-and-Bound-Verfahren mit Ausnahme von CBB für das Testset P10π dominiert wird. Der Performance-Vorteil der anderen Lösungsverfahren gegenüber BOT kann dabei insbesondere für das Schirmer-Alvarez-Valdes-Testset mit einer deutlichen Zunahme der Dominanz bei steigender Instanzgröße festgestellt werden. Weiterhin kann analog zum UBOπ-Testset PBB als das beste Lösungsverfahren herausgestellt wer-

		#nTriv	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$
P10π	PBB	2108	827	827	1281	0	0,004 s	0,002 s
	CBB		827	827	1281	0	0,056 s	0,054 s
	RBB		827	827	1281	0	0,007 s	0,007 s
	BOT		827	827	1281	0	0,023 s	0,023 s
P15π	PBB	204	188	188	16	0	0,127 s	0,001 s
	CBB		188	188	16	0	1,845 s	0,051 s
	RBB		188	188	16	0	2,114 s	0,002 s
	BOT		181	181	16	7	2,727 s	0,036 s
P20π	PBB	165	144	146	17	2	5,851 s	0,002 s
	CBB		139	142	17	6	0,112 s	0,052 s
	RBB		139	142	17	6	0,660 s	0,002 s
	BOT		136	139	16	10	3,974 s	2,187 s
P25π	PBB	136	113	120	14	2	0,111 s	0,039 s
	CBB		112	116	14	6	0,109 s	0,062 s
	RBB		112	115	14	7	0,018 s	0,010 s
	BOT		105	111	11	14	0,465 s	25,457 s
P30π	PBB	122	104	108	8	6	0,046 s	0,037 s
	CBB		104	104	8	10	0,072 s	0,053 s
	RBB		104	104	8	10	0,029 s	0,003 s
	BOT		98	104	3	15	2,095 s	0,196 s

Tabelle 7.12: Performance der Branch-and-Bound-Verfahren für das Böttcher-Testset

den, wobei der Performance-Vorteil von PBB gegenüber allen anderen Lösungsverfahren insbesondere aus den Ergebnissen für das Schirmer-Alvarez-Valdes-Testset hervorgeht. Während PBB für das Böttcher-Testset vor allem mehr zulässige Testinstanzen identifiziert, kann PBB für die Testsets mit bis zu 40 realen Vorgängen des Schirmer-Alvarez-Valdes-Testsets einen Großteil der Instanzen optimal lösen. Im Gegensatz zum UBOπ-Testset geht aus dem Vergleich zwischen CBB und RBB kein Verfahren als eindeutig dominant hervor. Lediglich für die Testsets J30π und J40π kann eine etwas bessere Performance von CBB festgestellt werden, wobei RBB tendenziell geringere Berechnungszeiten über alle Testsets aufweist.

Die Untersuchungen in diesem Abschnitt konnten zeigen, dass PBB am besten geeignet ist, die Instanzen aller betrachteten Testsets sowohl für das RCPSP/max-π als auch für das RCPSP/π exakt zu lösen. Weiterhin konnte festgestellt werden, dass PBB im Vergleich zu allen anderen Branch-and-Bound-Verfahren die beste Lösungsgüte über alle Testinstanzen des UBOπ-Testsets erzielt und darüber hinaus auch für die größte Anzahl an Testinstanzen den Lösungsstatus bestimmen kann. Die vorteilhafte Performance von PBB konnte insbesondere auf den Enumerationsansatz des Verfahrens zurückgeführt

		#nTriv	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$
J10π	PBB	808	803	803	5	0	0,047 s	0,041 s
	CBB		803	803	5	0	0,063 s	0,055 s
	RBB		803	803	5	0	0,060 s	0,052 s
	BOT		802	802	5	1	0,171 s	0,041 s
J20π	PBB	565	565	565	0	0	0,121 s	–
	CBB		563	565	0	0	0,906 s	–
	RBB		564	565	0	0	1,785 s	–
	BOT		509	561	0	4	6,436 s	–
J30π	PBB	453	452	453	0	0	0,727 s	–
	CBB		431	453	0	0	3,189 s	–
	RBB		427	453	0	0	3,717 s	–
	BOT		345	435	0	18	5,573 s	–
J40π	PBB	386	378	386	0	0	2,182 s	–
	CBB		347	386	0	0	5,386 s	–
	RBB		341	386	0	0	4,103 s	–
	BOT		261	363	0	23	4,376 s	–
J60π	PBB	346	314	346	0	0	3,621 s	–
	CBB		269	346	0	0	8,476 s	–
	RBB		268	346	0	0	2,172 s	–
	BOT		186	309	0	37	2,502 s	–

Tabelle 7.13: Performance der Branch-and-Bound-Verfahren für das Schirmer-Alvarez-Valdes-Testset

werden, wohingegen für alle anderen Branch-and-Bound-Verfahren ein entscheidender Einfluss der Verbesserungstechniken auf die Performance festgestellt wurde.

7.3 Vergleich mit zeitindexbasierten Modellformulierungen

Im Folgenden wird das PBB, für das in Abschnitt 7.2 gezeigt werden konnte, dass es alle anderen Branch-and-Bound-Verfahren über alle betrachteten Testsets dominiert, mit dem MILP-Solver IBM CPLEX (CPX) verglichen, der zur Lösung der zeitindexbasierten Modelle aus Kapitel 6 eingesetzt wird.

Tabelle 7.14 zeigt die Performance von PBB und der zeitindexbasierten Modellformulierungen aus Kapitel 6, die durch CPX gelöst wurden, für das UBOπ-Testset. In der Tabelle werden für den MILP-Solver CPX die Ergebnisse der aggregierten Pulse- (PA) und Step-Formulierung (SA), der disaggregierten Pulse- (PD) und Step-Formulierung

		#nTriv	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$
UBO10π	PBB	693	534	534	159	0	0,018 s	0,004 s
	PA		534	534	159	0	0,549 s	0,049 s
	PD		534	534	159	0	0,797 s	0,059 s
	SA		534	534	159	0	0,874 s	0,322 s
	SD		534	534	159	0	0,619 s	0,278 s
	OO		516	534	159	0	19,819 s	2,600 s
UBO20π	PBB	621	542	581	40	0	7,572 s	0,583 s
	PA		501	574	40	7	16,160 s	0,193 s
	PD		510	580	40	1	18,412 s	0,836 s
	SA		499	574	40	7	27,170 s	1,997 s
	SD		537	580	40	1	19,601 s	1,700 s
	OO		244	547	32	42	25,223 s	13,147 s
UBO50π	PBB	527	238	496	6	25	13,938 s	25,092 s
	PA		128	374	21	132	17,326 s	14,979 s
	PD		154	471	25	31	63,217 s	40,605 s
	SA		134	457	21	49	55,303 s	55,508 s
	SD		172	481	26	20	58,111 s	121,608 s
	OO		77	376	9	142	34,886 s	90,839 s
UBO100π	PBB	484	136	474	0	10	25,426 s	–
	PA		55	69	5	410	49,313 s	29,841 s
	PD		6	15	0	469	474,657 s	–
	SA		44	461	1	22	106,157 s	536,962 s
	SD		82	465	0	19	240,096 s	–
	OO		7	8	0	476	379,642 s	–
UBO200π	PBB	466	124	466	0	0	33,223 s	–
	PA		23	30	0	436	117,620 s	–
	PD		–	–	–	–	–	–
	SA		13	461	0	5	444,726 s	–
	SD		1	51	0	415	586,481 s	–
	OO		–	–	–	–	–	–

Tabelle 7.14: Performance von PBB und der zeitindexbasierten Modellformulierungen für das UBOπ-Testset

(SD) sowie der On/Off-Formulierung (OO) angegeben. Zunächst geht aus den Ergebnissen der Tabelle 7.14 hervor, dass PBB alle Modellformulierungen über alle Testsets im Hinblick auf die lösbaren Testinstanzen dominiert. Während PBB das Testset UBO10π lediglich schneller löst als alle zeitindexbasierten Modelle, kann PBB darüber hinaus für alle weiteren Testsets sowohl mehr lösbare Instanzen identifizieren als auch für mehr Testinstanzen eine optimale Lösung bestimmen und verifizieren. Eine bessere Performance zeigen die zeitindexbasierten Modelle dagegen in Bezug auf die Verifizierung unlösbarer Testinstanzen für die Testsets UBO50π und UBO100π. Dabei werden die besten Ergebnisse für UBO50π durch die disaggregierten Formulierungen PD und SD erzielt,

wohingegen unlösbare Testinstanzen für UBO100π nur noch durch die aggregierten Formulierungen PA und SA verifiziert werden können. PBB zeigt lediglich für die Testsets UBO10π und UBO20π eine vorteilhafte Performance in Bezug auf die Verifizierung unlösbarer Testinstanzen im Vergleich zu den Modellformulierungen, wobei nur PA eine kürzere durchschnittliche Berechnungszeit für die Bestimmung aller unlösbarer Instanzen für das Testset UBO20π aufweist. Durch einen Vergleich aller zeitindexbasierten Modellformulierungen kann SD als bestes Modell für alle Testsets mit bis zu 100 realen Vorgängen identifiziert werden, wohingegen OO die schlechteste Performance über alle Testsets zeigt. Weiterhin kann festgestellt werden, dass die vorteilhafte Performance der disaggregierten Formulierungen PD und SD, die zunächst im Vergleich zu den aggregierten Formulierungen PA und SA für Testsets mit kleineren Instanzen vorliegt, nicht mehr für größere Testinstanzen gegeben ist. Ein Grund dafür könnte sein, dass die Modellierung der Zeitrestriktionen in disaggregierter Form im Vergleich zu den aggregierten Modellen mehr Nebenbedingungen erfordert, sodass die disaggregierten Modelle schneller eine kritische Größe erreichen, die durch den Solver oder den physischen Speicher kaum oder gar nicht mehr verarbeitet werden kann. Dies würde auch erklären, weshalb CPX für die Modelle PD und OO nicht alle Instanzen des Testsets UBO200π ohne Fehlermeldung durchlaufen kann, was durch „–“ für alle Vergleichsgrößen der Formulierungen PD und OO in Tabelle 7.14 gekennzeichnet ist.

Im nächsten Schritt werden die Lösungsgüten von PBB und CPX für jede zeitindexbasierte Modellformulierung miteinander verglichen. Für die Untersuchungen werden die gleichen Bezeichner wie in den Tabellen 7.4 und 7.5 in Abschnitt 7.2 verwendet, wobei der einzige Unterschied darin besteht, dass nur zwei Lösungsverfahren betrachtet werden. Die Ergebnisse für den Vergleich zwischen PBB und CPX für jedes der zeitindexbasierten Modelle kann den Tabellen 7.15 bis 7.19 entnommen werden.

Testset	$\#^{\cup}_{\text{zul}}$	$\#^{\cap}_{\text{zul}}$	$\#^{\cup,\text{PBB}}_{\text{unb}}$	$\#^{\cup,\text{CPX}}_{\text{unb}}$	$\#^{\cap,\text{nv}}_{\text{zul}}$	$p^{\#,\text{PBB}}_{\text{best}}$	$p^{\#,\text{CPX}}_{\text{best}}$	$\varnothing^{\Delta,\text{PBB}}_{\text{best}}$	$\varnothing^{\Delta,\text{CPX}}_{\text{best}}$
UBO20π	581	574	0	7	81	95,06 %	40,74 %	0,03 %	6,42 %
UBO50π	496	374	0	122	251	98,01 %	3,59 %	0,47 %	33,74 %
UBO100π	474	69	0	405	15	93,33 %	6,67 %	0,09 %	196,44 %
UBO200π	466	30	0	436	7	100,00 %	0,00 %	0,00 %	526,80 %

Tabelle 7.15: Vergleich von PBB mit PA für das UBOπ-Testset

Testset	$\#^{\cup}_{\text{zul}}$	$\#^{\cap}_{\text{zul}}$	$\#^{\cup,\text{PBB}}_{\text{unb}}$	$\#^{\cup,\text{CPX}}_{\text{unb}}$	$\#^{\cap,\text{nv}}_{\text{zul}}$	$p^{\#,\text{PBB}}_{\text{best}}$	$p^{\#,\text{CPX}}_{\text{best}}$	$\varnothing^{\Delta,\text{PBB}}_{\text{best}}$	$\varnothing^{\Delta,\text{CPX}}_{\text{best}}$
UBO20π	581	580	0	1	78	94,87 %	41,03 %	0,39 %	5,33 %
UBO50π	497	470	1	26	322	97,20 %	6,21 %	0,52 %	28,58 %
UBO100π	474	15	0	459	9	100,00 %	44,44 %	0,00 %	56,51 %

Tabelle 7.16: Vergleich von PBB mit PD für das UBOπ-Testset

Testset	$\#^{\cup}_{\text{zul}}$	$\#^{\cap}_{\text{zul}}$	$\#^{\cup,\text{PBB}}_{\text{unb}}$	$\#^{\cup,\text{CPX}}_{\text{unb}}$	$\#^{\cap,\text{nv}}_{\text{zul}}$	$p^{\#,\text{PBB}}_{\text{best}}$	$p^{\#,\text{CPX}}_{\text{best}}$	$\varnothing^{\Delta,\text{PBB}}_{\text{best}}$	$\varnothing^{\Delta,\text{CPX}}_{\text{best}}$
UBO20π	581	574	0	7	81	97,53 %	45,68 %	0,02 %	6,55 %
UBO50π	496	457	0	39	325	99,38 %	1,23 %	0,15 %	45,83 %
UBO100π	474	461	0	13	417	100,00 %	0,00 %	0,00 %	116,96 %
UBO200π	466	461	0	5	448	100,00 %	0,00 %	0,00 %	169,46 %

Tabelle 7.17: Vergleich von PBB mit SA für das UBOπ-Testset

Testset	$\#^{\cup}_{\text{zul}}$	$\#^{\cap}_{\text{zul}}$	$\#^{\cup,\text{PBB}}_{\text{unb}}$	$\#^{\cup,\text{CPX}}_{\text{unb}}$	$\#^{\cap,\text{nv}}_{\text{zul}}$	$p^{\#,\text{PBB}}_{\text{best}}$	$p^{\#,\text{CPX}}_{\text{best}}$	$\varnothing^{\Delta,\text{PBB}}_{\text{best}}$	$\varnothing^{\Delta,\text{CPX}}_{\text{best}}$
UBO20π	581	580	0	1	58	84,48 %	63,79 %	1,00 %	6,07 %
UBO50π	496	481	0	15	315	94,92 %	12,06 %	0,49 %	25,86 %
UBO100π	474	465	0	9	390	98,21 %	1,79 %	0,26 %	51,08 %
UBO200π	466	51	0	415	50	100,00 %	0,00 %	0,00 %	83,44 %

Tabelle 7.18: Vergleich von PBB mit SD für das UBOπ-Testset

Testset	$\#^{\cup}_{\text{zul}}$	$\#^{\cap}_{\text{zul}}$	$\#^{\cup,\text{PBB}}_{\text{unb}}$	$\#^{\cup,\text{CPX}}_{\text{unb}}$	$\#^{\cap,\text{nv}}_{\text{zul}}$	$p^{\#,\text{PBB}}_{\text{best}}$	$p^{\#,\text{CPX}}_{\text{best}}$	$\varnothing^{\Delta,\text{PBB}}_{\text{best}}$	$\varnothing^{\Delta,\text{CPX}}_{\text{best}}$
UBO10π	534	534	0	0	18	100,00 %	22,22 %	0,00 %	10,55 %
UBO20π	581	547	0	34	303	100,00 %	3,63 %	0,00 %	22,89 %
UBO50π	496	376	0	120	299	100,00 %	0,00 %	0,00 %	79,82 %
UBO100π	474	8	0	466	1	100,00 %	0,00 %	0,00 %	283,33 %

Tabelle 7.19: Vergleich von PBB mit OO für das UBOπ-Testset

Die Auswertungen zeigen zunächst, dass lediglich eine einzige Instanz über alle Testsets und Modellformulierungen durch CPX als lösbar identifiziert wurde, für die PBB keine zulässige Lösung bestimmen konnte (s. Tabelle 7.16). Alle anderen Testinstanzen, für die durch CPX mit mindestens einer Modellformulierung eine zulässige Lösung ermittelt wurde, konnten dagegen durch PBB als lösbar identifiziert werden. Weiterhin geht aus den Ergebnissen in den Tabellen hervor, dass PBB im Vergleich zu den Modellformulierungen für einen überwiegenden Anteil aller betrachteten Testinstanzen ($\#^{\cap,\text{nv}}_{\text{zul}}$) eine Lösung mit kürzester Projektdauer bestimmt ($p^{\#,\text{PBB}}_{\text{best}}$). Darüber hinaus zeigen die letzten beiden Spalten der Tabellen, dass die durchschnittliche Güte der besten ermittelten Lösungen von PBB über alle betrachteten Testinstanzen ($\#^{\cap,\text{nv}}_{\text{zul}}$) die Güte aller Modellformulierungen, insbesondere für große Testinstanzen, deutlich dominiert.

Die Tabellen 7.20 und 7.21 zeigen die Ergebnisse der Untersuchungen von PBB sowie der zeitindexbasierten Modellformulierungen für das Böttcher- und das Schirmer-Alvarez-Valdes-Testset. Für das Böttcher-Testset werden zunächst durch PBB die besten Ergebnisse für P10π, P15π und P30π erzielt, wohingegen die Modellformulierungen SA und SD eine bessere Performance für P20π aufweisen. Für das Testset P25π liegt dagegen kein dominantes Lösungsverfahren vor. Während PBB im Vergleich zu allen Modellformulierungen für mehr Testinstanzen zulässige Lösungen ermittelt, werden durch SD

		#nTriv	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$
P10π	PBB	2108	827	827	1281	0	0,004 s	0,002 s
	PA		827	827	1281	0	0,061 s	0,032 s
	PD		827	827	1281	0	0,056 s	0,033 s
	SA		827	827	1281	0	0,057 s	0,038 s
	SD		827	827	1281	0	0,047 s	0,032 s
	OO		827	827	1281	0	0,089 s	0,054 s
P15π	PBB	204	188	188	16	0	0,127 s	0,001 s
	PA		188	188	16	0	0,566 s	0,027 s
	PD		188	188	16	0	1,012 s	0,031 s
	SA		188	188	16	0	0,244 s	0,034 s
	SD		188	188	16	0	0,132 s	0,027 s
	OO		188	188	16	0	4,371 s	0,036 s
P20π	PBB	165	144	146	17	2	5,851 s	0,002 s
	PA		143	145	17	3	4,052 s	0,030 s
	PD		145	145	17	3	7,603 s	0,038 s
	SA		145	146	17	2	1,720 s	0,030 s
	SD		145	146	17	2	1,084 s	0,030 s
	OO		139	140	17	8	1,494 s	0,059 s
P25π	PBB	136	113	120	14	2	0,111 s	0,039 s
	PA		114	115	14	7	3,041 s	0,034 s
	PD		113	115	14	7	0,234 s	0,054 s
	SA		115	115	14	7	1,314 s	0,035 s
	SD		116	119	14	3	1,423 s	0,036 s
	OO		112	112	14	10	0,724 s	0,088 s
P30π	PBB	122	104	108	8	6	0,046 s	0,037 s
	PA		104	104	8	10	0,179 s	0,031 s
	PD		104	105	8	9	0,252 s	0,051 s
	SA		104	104	8	10	0,196 s	0,032 s
	SD		104	106	8	8	0,103 s	0,032 s
	OO		104	104	8	10	1,439 s	0,076 s

Tabelle 7.20: Performance von PBB und der zeitindexbasierten Modellformulierungen für das Böttcher-Testset

mehr Instanzen optimal gelöst. Für das Schirmer-Alvarez-Valdes-Testset werden J10π und J20π durch PBB am schnellsten gelöst, wobei PBB auch für das Testset J30π die besten Ergebnisse erzielt. Für die Testsets mit mehr als 30 realen Vorgängen können dagegen bessere Ergebnisse durch CPX erzielt werden. Während CPX mit den Modellformulierungen PA und SD für die Testsets J40π und J60π mehr Instanzen optimal löst als PBB, führt die Formulierung PD dagegen nur für das Testset J40π zu etwas besseren Ergebnissen im Vergleich zu PBB. Analog zum UBOπ-Testset werden sowohl für das Böttcher- als auch für das Schirmer-Alvarez-Valdes-Testset für alle Instanzen mit bis zu 40 realen Vorgängen die besten Ergebnisse von CPX mit der Modellformulierung SD erzielt,

		#nTriv	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$
J10π	PBB	808	803	803	5	0	0,047 s	0,041 s
	PA		803	803	5	0	0,107 s	0,027 s
	PD		803	803	5	0	0,104 s	0,032 s
	SA		803	803	5	0	0,126 s	0,041 s
	SD		803	803	5	0	0,098 s	0,038 s
	OO		803	803	5	0	1,273 s	0,050 s
J20π	PBB	565	565	565	0	0	0,121 s	–
	PA		565	565	0	0	0,391 s	–
	PD		565	565	0	0	0,399 s	–
	SA		565	565	0	0	0,804 s	–
	SD		565	565	0	0	0,308 s	–
	OO		525	562	0	3	9,100 s	–
J30π	PBB	453	452	453	0	0	0,727 s	–
	PA		452	453	0	0	1,395 s	–
	PD		451	453	0	0	1,963 s	–
	SA		452	453	0	0	4,952 s	–
	SD		452	453	0	0	0,965 s	–
	OO		376	445	0	8	7,782 s	–
J40π	PBB	386	378	386	0	0	2,182 s	–
	PA		386	386	0	0	4,397 s	–
	PD		380	386	0	0	5,604 s	–
	SA		372	384	0	2	11,139 s	–
	SD		386	386	0	0	3,090 s	–
	OO		294	373	0	13	12,947 s	–
J60π	PBB	346	314	346	0	0	3,621 s	–
	PA		337	346	0	0	17,608 s	–
	PD		307	345	0	1	15,449 s	–
	SA		296	333	0	13	16,957 s	–
	SD		340	345	0	1	16,713 s	–
	OO		231	316	0	30	28,923 s	–

Tabelle 7.21: Performance von PBB und der zeitindexbasierten Modellformulierungen für das Schirmer-Alvarez-Valdes-Testset

wohingegen OO über alle Testsets die schlechteste Performance aufweist. Für das Testset J60π stellt dagegen PA die einzige Formulierung dar, durch die alle lösbaren Testinstanzen identifiziert werden können, wohingegen durch SD die meisten Testinstanzen optimal gelöst werden. Zusammenfassend geht aus den Ergebnissen der Tabellen 7.20 und 7.21 hervor, dass der MILP-Solver IBM CPLEX deutlich besser zur Lösung der Testinstanzen für das RCPSP/π geeignet ist.

Es konnte gezeigt werden, dass PBB im Vergleich zu allen betrachteten zeitindexbasierten Modellformulierungen am besten geeignet ist, die Testinstanzen des UBOπ-Testsets exakt zu lösen sowie zulässige Lösungen zu ermitteln. Unter den zeitindexbasierten Modellen

konnte SD als bestes Modell für alle Testsets UBO$n\pi$ mit bis zu 100 realen Vorgängen identifiziert werden, wohingegen SA die beste Performance für das Testset UBO200π erzielen konnte. Die vorteilhafte Performance von PBB konnte weiterhin für die Testsets P10π, P15π, P30π, J10π, J20π und J30π im Hinblick auf bessere Laufzeiten und die Identifizierung lösbarer Testinstanzen festgestellt werden. Alle weiteren Testsets konnten dagegen durch mindestens eine zeitindexbasierte Modellformulierung besser gelöst werden, wobei SD als bestes Modell über alle Testinstanzen des Böttcher- und Schirmer-Alvarez-Valdes-Testsets mit bis zu 40 realen Vorgängen identifiziert wurde. Insbesondere konnte gezeigt werden, dass die Modellformulierungen SD und PA am besten geeignet sind, um die Testsets J40π und J60π zu lösen.

7.4 Vergleich mit Heuristiken

Im letzten Abschnitt dieses Kapitels werden die in dieser Arbeit entwickelten Branch-and-Bound-Verfahren mit Heuristiken aus der Literatur verglichen. Da bislang noch keine Näherungsverfahren für das RCPSP/max-π veröffentlicht wurden, beschränken sich die nachfolgenden Untersuchungen auf heuristische Verfahren zum RCPSP/π. Im Folgenden werden für den Vergleich unterschiedliche Ausführungsvarianten des Scatter-Search- und des GRASP-Verfahrens aus Alvarez-Valdes et al. (2006, 2008) betrachtet, die den besten bislang bekannten heuristischen Verfahren aus der Literatur für das RCPSP/π entsprechen.

In Tabelle 7.22 sind die Ergebnisse des Performance-Vergleichs der unterschiedlichen Ausführungsvarianten des Scatter-Search- und des GRASP-Verfahrens, die Tabelle 11.7 in Alvarez-Valdes et al. (2015) entnommen wurden, für das Schirmer-Alvarez-Valdes-Testset dargestellt. In der dritten Spalte der Tabelle 7.22 wird zunächst für jedes Testset die Anzahl der lösbaren und nicht-trivialen Testinstanzen angegeben, die für die Untersuchungen aller Lösungsverfahren berücksichtigt wurden (#inst). Für jedes Testset und jede Ausführungsvariante des Scatter-Search- und des GRASP-Verfahrens werden jeweils drei Vergleichsgrößen betrachtet ($\#^{\neq}_{\text{opt}}$, $\varnothing^{\Delta}_{\text{opt}}$, $\varnothing^{\text{cpu}}_{\text{inst}}$). Die erste Vergleichsgröße gibt zunächst an, für wie viele der untersuchten Testinstanzen (#inst) das jeweilige Näherungsverfahren keine optimale bzw. keine beste Lösung bestimmen konnte ($\#^{\neq}_{\text{opt}}$). Dabei ist zu beachten, dass in Alvarez-Valdes et al. (2015) für eine Instanz des Testsets J30π und für fünf Instanzen des Testsets J40π keine optimalen Lösungen verifiziert werden konnten, sodass für diese Instanzen lediglich die besten bekannten Lösungen für die Auswertungen betrachtet wurden. Zur Bestimmung der optimalen Lösungen wurde eine zeitindexbasierte Modellformulierung für das RCPSP/π verwendet, die mit dem MILP-Solver IBM

CPLEX gelöst wurde. Die weiteren Vergleichsgrößen geben die durchschnittliche Laufzeit ($\varnothing^{\text{cpu}}_{\text{inst}}$) sowie die durchschnittliche Abweichung der Projektdauer einer besten ermittelten Lösung vom Zielfunktionswert einer optimalen bzw. einer besten bekannten Lösung ($\varnothing^{\Delta}_{\text{opt}}$) über alle betrachteten Testinstanzen (#inst) für jedes Näherungsverfahren an. Durch „Regen 0“, „Regen 1“ und „Regen 6“ werden die unterschiedlichen Varianten des Scatter-Search-Verfahrens gekennzeichnet, die sich darin unterscheiden, ob und wie die Referenzmenge am Ende jeder Iteration neu bestimmt wird (engl. Regeneration). Für eine detaillierte Beschreibung des Scatter-Search-Verfahrens sei auf Alvarez-Valdes et al. (2006) verwiesen. Die Auswertungen für das GRASP-Verfahren beziehen sich auf eine „klassische“ (GRASP, GR+PR) und eine „aggressive“ (AG-GR, AG+PR) Form der Ausführung, wobei für die „aggressive“ Variante wiederholt ein Preprocessing-Verfahren in Kombination mit der Konstruktions- und Verbesserungsphase der „klassischen“ Variante angewendet wird. Weiterhin unterscheiden sich die Ausführungsvarianten dadurch, dass nach Abschluss des GRASP-Algorithmus entweder ein Path-Relinking-Verfahren (PR) ausgeführt wird (GR+PR, AG+PR) oder nicht (GRASP, AG-GR). Eine detaillierte Beschreibung des GRASP-Verfahrens kann der Arbeit Alvarez-Valdes et al. (2008) entnommen werden.

			Scatter-Search			GRASP			
		#inst	Regen 0	Regen 1	Regen 6	GRASP	GR+PR	AG-GR	AG+PR
J10π	$\#^{\neq}_{\text{opt}}$	803	3	0	0	1	1	1	1
	$\varnothing^{\Delta}_{\text{opt}}$		0,02 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %	0,00 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		1,2 s	1,8 s	2,5 s	0,9 s	0,9 s	0,3 s	0,3 s
J20π	$\#^{\neq}_{\text{opt}}$	565	22	10	5	43	32	22	19
	$\varnothing^{\Delta}_{\text{opt}}$		0,15 %	0,03 %	0,04 %	0,40 %	0,33 %	0,13 %	0,12 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		3,0 s	3,4 s	17,0 s	1,4 s	1,4 s	1,1 s	1,2 s
J30π	$\#^{\neq}_{\text{opt}}$	453	41	29	21	68	63	35	33
	$\varnothing^{\Delta}_{\text{opt}}$		0,32 %	0,19 %	0,10 %	1,00 %	0,88 %	0,24 %	0,21 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		7,2 s	11,8 s	28,8 s	2,9 s	3,1 s	3,4 s	3,7 s
J40π	$\#^{\neq}_{\text{opt}}$	386	61	41	31	89	84	59	54
	$\varnothing^{\Delta}_{\text{opt}}$		0,59 %	0,36 %	0,25 %	2,03 %	1,82 %	0,67 %	0,59 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		15,7 s	25,8 s	51,1 s	5,7 s	6,2 s	2,9 s	7,2 s
J60π	$\#^{\neq}_{\text{opt}}$	346	85	74	67	110	105	91	80
	$\varnothing^{\Delta}_{\text{opt}}$		1,22 %	0,90 %	0,71 %	3,68 %	3,31 %	1,38 %	1,16 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		55,8 s	105,2 s	175,9 s	8,7 s	10,3 s	10,6 s	13,4 s

Tabelle 7.22: Vergleich der Ausführungsvarianten des Scatter-Search- und des GRASP-Verfahrens aus Alvarez-Valdes et al. (2006, 2008) für das Schirmer-Alvarez-Valdes-Testset (Ergebnisse aus Tabelle 11.7 in Alvarez-Valdes et al. (2015))

Die unterschiedlichen Ausführungsvarianten des Scatter-Search- sowie des GRASP-Verfahrens wurden in C++ kodiert und auf einem Pentium IV Prozessor mit 2,8 GHz aus-

geführt (vgl. Alvarez-Valdes et al., 2015). Um für den nachfolgenden Vergleich mit den Branch-and-Bound-Verfahren die unterschiedlichen Taktfrequenzen der verwendeten Prozessoren zu berücksichtigen, wurden die gemessenen Laufzeiten der Branch-and-Bound-Verfahren ($\varnothing^{\text{cpu}}_{\text{inst}}$) mit einem Skalierungsfaktor von 3200/2800 = 8/7 multipliziert. Für die Untersuchungen der Branch-and-Bound-Verfahren wurde zunächst das gesamte Schirmer-Alvarez-Valdes-Testset durch den MILP-Solver IBM CPLEX mit der Modellformulierung PA im Multi-Thread-Modus vollständig gelöst, sodass sich alle nachfolgenden Auswertungen auf optimale Lösungen beziehen. Alle Branch-and-Bound-Verfahren wurden mit den gleichen Spezifikationen wie für die Untersuchungen in Abschnitt 7.2 ausgeführt.

In Tabelle 7.23 sind die Ergebnisse der Untersuchungen von PBB in Abhängigkeit verschiedener Zeitlimits[6] für das Schirmer-Alvarez-Valdes-Testset dargestellt, wobei durch „–“ angegeben wird, dass PBB bereits vor Erreichen des jeweiligen Zeitlimits alle optimalen Lösungen bestimmen und verifizieren konnte.

			Zeitlimit					
		#inst	10 s	30 s	60 s	100 s	200 s	300 s
J10π	$\#^{\neq}_{\text{opt}}$	803	0	–	–	–	–	–
	$\varnothing^{\Delta}_{\text{opt}}$		0,000 %	–	–	–	–	–
	$\varnothing^{\text{cpu}}_{\text{inst}}$		0,054 s	–	–	–	–	–
J20π	$\#^{\neq}_{\text{opt}}$	565	0	0	0	–	–	–
	$\varnothing^{\Delta}_{\text{opt}}$		0,000 %	0,000 %	0,000 %	–	–	–
	$\varnothing^{\text{cpu}}_{\text{inst}}$		0,086 s	0,127 s	0,139 s	–	–	–
J30π	$\#^{\neq}_{\text{opt}}$	453	3	1	1	1	1	1
	$\varnothing^{\Delta}_{\text{opt}}$		0,046 %	0,010 %	0,002 %	0,002 %	0,002 %	0,002 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		0,327 s	0,600 s	0,903 s	1,075 s	1,328 s	1,586 s
J40π	$\#^{\neq}_{\text{opt}}$	386	14	10	6	5	4	4
	$\varnothing^{\Delta}_{\text{opt}}$		0,109 %	0,058 %	0,030 %	0,028 %	0,025 %	0,022 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		0,711 s	1,754 s	2,899 s	4,108 s	6,975 s	9,548 s
J60π	$\#^{\neq}_{\text{opt}}$	346	41	37	33	29	28	28
	$\varnothing^{\Delta}_{\text{opt}}$		0,969 %	0,553 %	0,489 %	0,430 %	0,375 %	0,288 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		1,748 s	4,676 s	8,573 s	13,249 s	24,484 s	35,465 s

Tabelle 7.23: Lösungsgüte von PBB in Abhängigkeit eines vorgegebenen Zeitlimits für das Schirmer-Alvarez-Valdes-Testset

Aus den Ergebnissen in Tabelle 7.23 geht zunächst hervor, dass durch PBB mit einem Zeitlimit von 10 Sekunden für alle Testsets mit bis zu 40 realen Vorgängen und mit einem Zeitlimit von 30 Sekunden für das Testset J60π bereits deutlich bessere Ergebnisse

[6] Die vorgegebenen Zeitlimits für die Untersuchungen der Branch-and-Bound-Verfahren wurden im Gegensatz zu den gemessenen Laufzeiten ($\varnothing^{\text{cpu}}_{\text{inst}}$) nicht skaliert.

erzielt werden als durch alle Ausführungsvarianten des Scatter-Search- und des GRASP-Verfahrens. Es ist dabei insbesondere zu beachten, dass die hohen durchschnittlichen Laufzeiten der Ausführungsvarianten „Regen 1“ und „Regen 6“ des Scatter-Search-Verfahrens, die die besten Lösungsgüten über alle Näherungsverfahren erzielen, durch PBB weit unterschritten werden.

Die Tabellen 7.24 und 7.25 zeigen die Ergebnisse der Untersuchungen von CBB und RBB. Durch „–“ wird vor der ersten Eintragung für jedes Testset angegeben, dass CBB oder RBB nicht für alle betrachteten Testinstanzen (#inst) innerhalb des angegebenen Zeitlimits eine zulässige Lösung ermitteln konnte.

			Zeitlimit					
		#inst	10 s	30 s	60 s	100 s	200 s	300 s
J10π	$\#^{\neq}_{\text{opt}}$	803	0	–	–	–	–	–
	$\varnothing^{\Delta}_{\text{opt}}$		0,000 %	–	–	–	–	–
	$\varnothing^{\text{cpu}}_{\text{inst}}$		0,072 s	–	–	–	–	–
J20π	$\#^{\neq}_{\text{opt}}$	565	11	6	3	2	2	2
	$\varnothing^{\Delta}_{\text{opt}}$		0,162 %	0,070 %	0,054 %	0,036 %	0,022 %	0,022 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		0,442 s	0,845 s	1,168 s	1,435 s	1,834 s	2,245 s
J30π	$\#^{\neq}_{\text{opt}}$	453	37	30	24	22	21	18
	$\varnothing^{\Delta}_{\text{opt}}$		1,381 %	0,935 %	0,750 %	0,641 %	0,558 %	0,475 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		1,318 s	3,146 s	5,337 s	7,971 s	14,282 s	20,118 s
J40π	$\#^{\neq}_{\text{opt}}$	386	–	53	46	43	39	37
	$\varnothing^{\Delta}_{\text{opt}}$		–	2,880 %	2,329 %	1,970 %	1,670 %	1,476 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		–	5,506 s	10,101 s	15,614 s	28,192 s	40,174 s
J60π	$\#^{\neq}_{\text{opt}}$	346	–	83	79	76	74	72
	$\varnothing^{\Delta}_{\text{opt}}$		–	4,415 %	3,823 %	3,280 %	2,753 %	2,504 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		–	13,686 s	21,750 s	32,244 s	58,079 s	83,833 s

Tabelle 7.24: Lösungsgüte von CBB in Abhängigkeit eines vorgegebenen Zeitlimits für das Schirmer-Alvarez-Valdes-Testset

Im Gegensatz zu PBB können durch CBB und RBB lediglich für die Testsets J10π und J20π bessere Ergebnisse erzielt werden als durch die Ausführungsvarianten des Scatter-Search- und des GRASP-Verfahrens. Für größere Testinstanzen werden dagegen bereits durch AG-GR und AG+PR bessere Lösungsgüten nach deutlich geringeren Laufzeiten erzielt, die weder durch CBB noch RBB bei einem Zeitlimit von 300 Sekunden erreicht werden können.

Die Untersuchungen für das Böttcher-Testset konnten zeigen, dass durch einen Preprocessing-Schritt, der allen betrachteten Näherungsverfahren vorangestellt ist, bereits mehr lösbare Instanzen für die Testsets P25π und P30π identifiziert werden als durch jedes der Branch-and-Bound-Verfahren mit einem Zeitlimit von 300 Sekunden. Weiterhin konn-

			Zeitlimit					
		#inst	10 s	30 s	60 s	100 s	200 s	300 s
J10π	$\#^{\neq}_{\text{opt}}$	803	0	–	–	–	–	–
	$\varnothing^{\Delta}_{\text{opt}}$		0,000 %	–	–	–	–	–
	$\varnothing^{\text{cpu}}_{\text{inst}}$		0,069 s	–	–	–	–	–
J20π	$\#^{\neq}_{\text{opt}}$	565	11	9	4	3	1	1
	$\varnothing^{\Delta}_{\text{opt}}$		0,091 %	0,073 %	0,029 %	0,026 %	0,014 %	0,014 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		0,428 s	0,895 s	1,465 s	1,909 s	2,407 s	2,643 s
J30π	$\#^{\neq}_{\text{opt}}$	453	38	33	29	27	23	23
	$\varnothing^{\Delta}_{\text{opt}}$		1,083 %	0,723 %	0,585 %	0,480 %	0,370 %	0,339 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		1,247 s	3,160 s	5,768 s	9,001 s	16,667 s	23,683 s
J40π	$\#^{\neq}_{\text{opt}}$	386	–	–	49	46	43	42
	$\varnothing^{\Delta}_{\text{opt}}$		–	–	2,214 %	2,007 %	1,739 %	1,418 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		–	–	9,782 s	15,929 s	30,435 s	44,113 s
J60π	$\#^{\neq}_{\text{opt}}$	346	–	–	–	–	–	73
	$\varnothing^{\Delta}_{\text{opt}}$		–	–	–	–	–	6,492 %
	$\varnothing^{\text{cpu}}_{\text{inst}}$		–	–	–	–	–	79,214 s

Tabelle 7.25: Lösungsgüte von RBB in Abhängigkeit eines vorgegebenen Zeitlimits für das Schirmer-Alvarez-Valdes-Testset

ten durch die Branch-and-Bound-Verfahren lediglich für die Testsets P10π und P15π bessere Ergebnisse als durch die Näherungsverfahren erzielt werden. Entsprechend sollte zukünftig untersucht werden, inwieweit die Performance der betrachteten Branch-and-Bound-Verfahren durch das Preprocessing aus Alvarez-Valdes et al. (2006, 2008) für das Böttcher-Testset verbessert werden kann.

Zusammenfassend geht aus den Untersuchungen in diesem Abschnitt hervor, dass durch PBB bessere Ergebnisse für das Schirmer-Alvarez-Valdes-Testset erzielt werden als durch die besten bekannten Näherungsverfahren für das RCPSP/π, wohingegen durch die heuristischen Ansätze bessere Ergebnisse für das Böttcher-Testset erreicht werden. Weiterhin wurde gezeigt, dass sowohl CBB als auch RBB lediglich für Testinstanzen mit bis zu 20 realen Vorgängen Performance-Vorteile gegenüber den Näherungsverfahren aufweisen. Abschließend sei erwähnt, dass die Ergebnisse der Untersuchungen in diesem Abschnitt insbesondere nahelegen, dass für die zukünftige Entwicklung heuristischer Verfahren sowohl für das RCPSP/π als auch für das RCPSP/max-π Schedule-Generierungsschemata berücksichtigt werden sollten, die auf dem Partitionsansatz basieren.

Kapitel 8

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde das Projektdauerminimierungsproblem mit allgemeinen Zeitbeziehungen und partiell erneuerbaren Ressourcen (RCPSP/max-π) untersucht. Ein Schwerpunkt lag dabei auf der Entwicklung von Branch-and-Bound-Verfahren, die auf unterschiedlichen Enumerationsschemata basieren. Insgesamt wurden zwei relaxationsbasierte und ein konstruktionsbasiertes Branch-and-Bound-Verfahren vorgestellt. Für die relaxationsbasierten Ansätze wurden Enumerationsschemata entwickelt, die den Suchraum des Problems schrittweise durch das Hinzufügen von Schrankenwerten für die Ressourcenbelegungen der Vorgänge des Projekts einschränken. Weiterhin wurde für das konstruktionsbasierte Lösungsverfahren ein Ansatz beschrieben, der die Vorgänge des Projekts sukzessiv über den Enumerationsverlauf in Verbindung mit einem Ausplanungsschritt einplant. Für alle Lösungsverfahren wurde ein Projektplanungsproblem ohne Kapazitätsrestriktionen unter Einschränkung der Startzeitpunkte der Vorgänge eingeführt, das in jedem Enumerationsknoten der Branch-and-Bound-Verfahren gelöst wird. Für das entsprechende Projektplanungsproblem wurden Zeitplanungsverfahren, Konsistenztests sowie ein Verfahren zur Bestimmung einer destruktiven unteren Schranke für die kürzeste Projektdauer vorgestellt. Ferner wurden für zwei der Enumerationsansätze Verfahren entwickelt, die zur Vermeidung von Redundanzen im Enumerationsverlauf beitragen.

Ein weiterer Schwerpunkt dieser Arbeit war die Untersuchung des Zusammenhangs zwischen partiell erneuerbaren Ressourcen und anderen Modellierungsansätzen der Projektplanung. Es konnte festgestellt werden, dass unterschiedliche Modellierungskonzepte der Projektplanung, wie beispielsweise kumulative Ressourcen oder Kalender, durch partiell erneuerbare Ressourcen in Verbindung mit zusätzlichen Vorgängen und Zeitbeziehungen dargestellt werden können, wenn davon ausgegangen wird, dass jeder Vorgang des Projekts nur zu ganzzahligen Zeitpunkten starten kann und eine maximale Projektdauer

vorgegeben ist. Basierend darauf und unter Bezugnahme auf die strukturellen Eigenschaften der zugehörigen Projektplanungsprobleme konnte schließlich abgeleitet werden, dass eine kürzeste Projektdauer für jedes der Planungsprobleme durch ein exaktes Verfahren für das RCPSP/max-π oder für das RCPSP/π bestimmt werden kann.

Zur Einordnung der Performance der entwickelten Branch-and-Bound-Verfahren im Vergleich zu einem MILP-Solver wurden zudem ganzzahlige lineare Programme für das RCPSP/max-π vorgestellt, die aus zeitindexbasierten Modellformulierungen aus der Literatur zur Projektplanung abgeleitet wurden.

Durch eine experimentelle Performance-Analyse wurde gezeigt, dass das relaxationsbasierte (partitionsbasierte) Branch-and-Bound-Verfahren, das in jedem Verzweigungsschritt den Wertebereich der Startzeitpunkte eines der Vorgänge des Projekts partitioniert, am besten für die exakte Lösung und für die Bestimmung zulässiger Lösungen für das RCPSP/max-π geeignet ist. Lediglich für den Beweis unlösbarer Testinstanzen konnten durch den MILP-Solver IBM CPLEX für einige der zeitindexbasierten Modellformulierungen bessere Ergebnisse erzielt werden. Aus den Untersuchungen für das RCPSP/π ging weiterhin hervor, dass alle entwickelten Branch-and-Bound-Verfahren der vorliegenden Arbeit eine bessere Performance im Vergleich zum einzigen bekannten Branch-and-Bound-Algorithmus aus der Literatur für das RCPSP/π aufweisen. Wie auch für das RCPSP/max-π konnten die Ergebnisse zeigen, dass das partitionsbasierte Lösungsverfahren gegenüber allen anderen Branch-and-Bound-Verfahren am besten zur Lösung des RCPSP/π geeignet ist. Ein Vergleich mit dem MILP-Solver IBM CPLEX konnte dagegen aufzeigen, dass für größere Testinstanzen mit 40 und 60 realen Vorgängen durch einige Modellformulierungen bessere Ergebnisse erzielt werden können als durch alle Branch-and-Bound-Verfahren. Durch einen abschließenden Vergleich konnte darüber hinaus für ein Testset des RCPSP/π festgestellt werden, dass das partitionsbasierte Branch-and-Bound-Verfahren in kürzerer Zeit bessere Lösungsgüten erreicht als die besten bekannten Näherungsverfahren für das RCPSP/π, wohingegen für alle anderen Branch-and-Bound-Verfahren diese vorteilhafte Performance nicht ermittelt werden konnte. Für ein weiteres Testset des RCPSP/π konnten dagegen durch die entwickelten Branch-and-Bound-Verfahren keine besseren Ergebnisse im Vergleich zu den Näherungsverfahren erzielt werden.

Aufbauend auf den in der vorliegenden Arbeit gelegten Grundlagen zur Projektplanung mit partiell erneuerbaren Ressourcen kommen unter anderem die folgenden Themenbereiche für zukünfige Untersuchungen infrage.

Basierend auf den Ergebnissen der experimentellen Performance-Analyse, aus denen unter anderem hervorgeht, dass Konsistenztests entscheidend zur Verbesserung aller in dieser Arbeit entwickelten Branch-and-Bound-Verfahren beitragen, wäre eine wichtige Forschungsaufgabe die Entwicklung und Untersuchung weiterer Konsistenztests. Ein in diesem Zusammenhang ebenfalls interessantes Thema wäre die Untersuchung der Performance von CP-Solvern (Constraint Programming), die sowohl zur Lösung der zeitindexbasierten Modellformulierungen aus Kapitel 6 als auch für die Lösung von CP-Modellen eingesetzt werden könnten. Zukünftig sollten dementsprechend auch CP-Modelle für das RCPSP/max-π formuliert und untersucht werden.

Ein weiteres relevantes Thema ist die Untersuchung der Performance der in dieser Arbeit entwickelten Branch-and-Bound-Verfahren für Probleme mit beliebiger regulärer Zielfunktion. Dazu wären lediglich die oberen und unteren Schrankenwerte und alle mit ihnen verbundenen Ausführungsschritte an die jeweils betrachtete Zielfunktion anzupassen. Dabei ist insbesondere zu beachten, dass die unteren Schranken $LB0^{\pi} := ES_{n+1}(W)$ und LBD^{π} im Allgemeinen nicht einsetzbar sind, wohingegen die Konsistenztests und Verfahren zur Redundanzvermeidung unabhängig von der betrachteten regulären Zielfunktion weiterhin ausgeführt werden können.

Die in dieser Arbeit entwickelten Branch-and-Bound-Verfahren können wie für reguläre Zielfunktionen auch zur Lösung von Problemen mit beliebigen antiregulären Zielfunktionen eingesetzt werden. Dafür sind zunächst die gleichen Anpassungen wie für die regulären Zielfunktionen erforderlich. Darüber hinaus wird der Initialisierungsschritt für jedes Lösungsverfahren angepasst, indem der Schedule S des Wurzelknotens durch $S := \max \mathcal{S}_T(W)$ festgelegt wird. Weiterhin werden in jedem Iterationsschritt der Relaxationsansätze die Schedules durch $S := \max \mathcal{S}_T(W)$ bestimmt, wobei ansonsten keine weiteren Anpassungen erforderlich sind. Für den Konstruktionsansatz wird der Verzweigungsschritt durch $S'_i := t$, $S' := \max \widehat{\mathcal{S}}_T(W, i, S'_i)$ und $W' := (W_j \backslash]S'_j, \infty[)_{j \in V}$ ersetzt und für die Berechnung von T_i ein Verfahren eingesetzt, das in Anlehnung an Algorithmus 5.7 die Zeitpunkte in Θ_i in umgekehrter Reihenfolge durchläuft. Weitere Anpassungen für den Konstruktionsansatz betreffen den Ausplanungsschritt, die Redundanzvermeidungstechnik ULT und die Aktualisierung von T_i, nachdem eine neue beste Lösung gefunden wurde.

Wie dargelegt wurde, können alle in dieser Arbeit entwickelten Branch-and-Bound-Verfahren durch wenige Anpassungen für Probleme mit regulären und antiregulären Zielfunktionen eingesetzt werden. Für weitere Zielfunktionsklassen scheinen dagegen nur die relaxationsbasierten Enumerationsansätze geeignet zu sein, wohingegen der konstruktionsbasierte Ansatz ohne die Berechnung von T_i bzw. einer anderen zielspezifischen Einschrän-

kung vermutlich keine gute Performance erzielen kann. Die beiden relaxationsbasierten Enumerationsschemata können in Verbindung mit den entwickelten Konsistenztests und Verfahren zur Redundanzvermeidung für die Optimierung beliebiger Zielfunktionen eingesetzt werden, sofern ein Verfahren existiert, das die Zielfunktion auf der Menge $\mathcal{S}_T(W)$ minimiert. Interessant wäre in diesem Zusammenhang die Bestimmung der Komplexitätsklassen für unterschiedliche Problemstellungen $\min\{f(S) \mid S \in \mathcal{S}_T(W)\}$ mit einer im Hinblick auf partiell erneuerbare Ressourcen geeigneten Zielfunktion $f : \mathbb{Z}_{\geq 0}^{n+2} \mapsto \mathbb{R}$. Aufbauend darauf könnte dann entschieden werden, ob entweder Relaxationsansätze zur Lösung der Problemstellungen geeignet sind oder konstruktionsbasierte Verfahren entwickelt werden sollten. Ein sinnvoller Einsatz der Relaxationsansätze kann dabei insbesondere dann vermutet werden, falls polynomielle oder pseudopolynomielle Verfahren zur Lösung des Problems $\min\{f(S) \mid S \in \mathcal{S}_T(W)\}$ bekannt sind. In Anlehnung an Neumann et al. (2003, Kapitel 3) stellt in diesem Zusammenhang auch die Kategorisierung verschiedener Zielfunktionen sowie die Untersuchung der strukturellen Eigenschaften der zugehörigen Problemstellungen eine weitere wichtige Forschungsaufgabe dar.

Die partiell erneuerbaren Ressourcen werden in der Literatur sowie in der vorliegenden Arbeit vor allem durch die Möglichkeit der Modellierung von Arbeitszeitvereinbarungen motiviert. Dabei wird die Verfügbarkeit des Personals in jeder einzelnen Zeitperiode durch partiell erneuerbare Ressourcen anstatt durch erneuerbare Ressourcen abgebildet. Da für Instanzen mit partiell erneuerbaren Ressourcen, die auch erneuerbare Ressourcen enthalten, eine bessere Performance durch Lösungsverfahren zu erwarten ist, die zusätzlich die spezifischen Eigenschaften von erneuerbaren Ressourcen berücksichtigen, wäre die Entwicklung solcher Lösungsverfahren eine interessante Forschungsaufgabe. In diesem Zusammenhang wäre es auch interessant zu untersuchen, inwieweit sich die Performance dieser Lösungsverfahren von den in dieser Arbeit entwickelten Branch-and-Bound-Verfahren für Instanzen mit erneuerbaren Ressourcen unterscheidet. Für die Entwicklung exakter Lösungsverfahren wäre beispielsweise die Kombination der relaxationsbasierten Branch-and-Bound-Verfahren in dieser Arbeit mit den Enumerationsschemata aus De Reyck und Herroelen (1998) oder Fest et al. (1999) denkbar.

Die Erweiterung um mehrere Ausführungsmodi der Vorgänge eines Projekts ist bei partiell erneuerbaren Ressourcen wie auch bei anderen Ressourcentypen zur Modellierung realer Anwendungen generell sinnvoll. Beispielsweise kann dadurch ein Trade-off zwischen der Ausführungsdauer und dem Ressourcenbedarf eines Vorgangs oder alternative Bearbeitungsmöglichkeiten abgebildet werden. Im Folgenden wird ein weiterer für partiell erneuerbare Ressourcen spezifischer Aspekt für den Einsatz mehrerer Ausführungsmodi diskutiert, der die „aggregierte“ Modellierung mehrerer gleichartiger Ressourcen

betrifft. Während für erneuerbare Ressourcen die Kapazitätsrestriktionen mehrerer einzelner Ressourceneinheiten (z. B. Maschinen) in jeder Zeitperiode „aggregiert“ dargestellt werden können, indem die Kapazitäten aller Ressourcen in einer Zeitperiode summiert werden, ist diese Art der Modellierung für partiell erneuerbare Ressourcen nicht möglich. Durch das Beispiel in Abbildung 8.1 wird im Folgenden gezeigt, dass die „aggregierte“ Modellierung gleichartiger Ressourcen (z. B. Arbeitskräfte mit gleichen Arbeitszeitvereinbarungen) durch partiell erneuerbare Ressourcen nicht umsetzbar ist. In Anlehnung an das Anwendungsbeispiel in Abschnitt 2.2 wird angenommen, dass zwei Kundenbetreuer (K1, K2) einem Projekt zugewiesen sind, die an einem Wochenende freigestellt werden, falls sie am Wochenende zuvor sowohl am Samstag als auch am Sonntag gearbeitet haben. Es wird dabei ein Planungshorizont über drei Wochenenden betrachtet. Analog zu Abschnitt 2.2 könnte die Arbeitszeitvereinbarung jedes Kundenbetreuers durch die partiell erneuerbaren Ressourcen $\Pi_1 = \{1, 2, 8\}$, $\Pi_2 = \{1, 2, 9\}$, $\Pi_3 = \{8, 9, 15\}$ und $\Pi_4 = \{8, 9, 16\}$ mit Kapazitäten von $R_1 = R_2 = R_3 = R_4 = 2$ abgebildet werden. Die „aggregierte“ Modellierung der Arbeitszeitvereinbarungen beider Kundenbetreuer wäre dann durch die Summe der Kapazitäten mit $R_1 = R_2 = R_3 = R_4 = 4$ gegeben. Abbildung 8.1 zeigt eine mögliche Belegung der beiden Kundenbetreuer K1 und K2 (auf jeweils einer Ebene) durch Vorgänge, die jeweils einen Ressourcenbedarf von einer Einheit haben. Zunächst kann festgestellt werden, dass die dargestellte Belegung der Ressourcen für die „aggregierte“ Modellierung zulässig ist, wohingegen die Arbeitszeitvereinbarung für mindestens einen der Kundenbetreuer, unabhängig von der Arbeitsverteilung in jeder einzelnen Zeitperiode, verletzt wird. Das Beispiel in Abbildung 8.1 verdeutlicht, dass ein

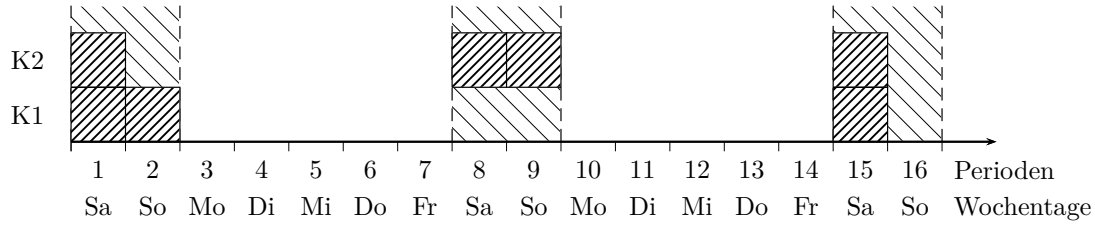

Abbildung 8.1: „Aggregierte“ Modellierung partiell erneuerbarer Ressourcen

„Pool“ mehrerer gleichartiger Ressourcen durch partiell erneuerbare Ressourcen nicht modelliert werden kann. Entsprechend kann die Problemstellung, die in dieser Arbeit betrachtet wird, nicht dazu verwendet werden, um Vorgänge einem „Pool“ gleichartiger Ressourcen zuzuordnen. Stattdessen ist vor dem Einplanungsschritt der Vorgänge eine Zuordnung der Vorgänge zu den einzelnen Ressourcen aus einem „Pool“ erforderlich, was eine erhebliche Einschränkung für die Modellierung praktischer Anwendungen sein kann. Die Erweiterung um mehrere Ausführungsmodi der Vorgänge eines Projekts mit partiell

erneuerbaren Ressourcen scheint daher die einzige Möglichkeit zu sein, um den Vorgängen mehrere gleichartige Ressourcen als alternative Bearbeitungsmöglichkeiten zuordnen zu können. Eine weitere Schlussfolgerung, die aus dem beschriebenen Sachverhalt gezogen werden kann, ist zudem, dass der Spezialfall unärer partiell erneuerbarer Ressourcen näher betrachtet werden sollte, um dadurch eventuell effektivere Konsistenztests und untere Schranken zu entwickeln.

Literaturverzeichnis

Ahuja, R. K., T. L. Magnanti und J. B. Orlin (1993). *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs: Prentice-Hall. (Zitiert auf den Seiten 79 und 89)

Alvarez-Valdes, R., E. Crespo, J. M. Tamarit und F. Villa (2006). A scatter search algorithm for project scheduling under partially renewable resources. *Journal of Heuristics 12*(1), 95–113. (Zitiert auf den Seiten 15, 52, 57, 100, 130, 131, 149, 150 und 153)

Alvarez-Valdes, R., E. Crespo, J. M. Tamarit und F. Villa (2008). GRASP and path relinking for project scheduling under partially renewable resources. *European Journal of Operational Research 189*(3), 1153–1170. (Zitiert auf den Seiten 15, 100, 135, 149, 150 und 153)

Alvarez-Valdes, R., J. M. Tamarit und F. Villa (2015). Partially renewable resources. In: C. Schwindt und J. Zimmermann (Hg.), *Handbook on Project Management and Scheduling Vol.1*, 203–227. Cham: Springer. (Zitiert auf den Seiten 16, 100, 149, 150 und 151)

Androutsopoulos, K. N., E. G. Manousakis und M. A. Madas (2020). Modeling and solving a bi-objective airport slot scheduling problem. *European Journal of Operational Research 284*(1), 135–151. (Zitiert auf Seite 14)

Artigues, C. (2017). On the strength of time-indexed formulations for the resource-constrained project scheduling problem. *Operations Research Letters 45*(2), 154–159. (Zitiert auf Seite 121)

Artigues, C., O. Koné, P. Lopez und M. Mongeau (2015). Mixed-integer linear programming formulations. In: C. Schwindt und J. Zimmermann (Hg.), *Handbook on Project Management and Scheduling Vol.1*, 17–41. Cham: Springer. (Zitiert auf Seite 121)

Bartsch, T., A. Drexl und S. Kröger (2006). Scheduling the professional soccer leagues of Austria and Germany. *Computers & Operations Research 33*(7), 1907–1937. (Zitiert auf Seite 12)

Bartusch, M., R. H. Möhring und F. J. Radermacher (1988). Scheduling project networks with resource constraints and time windows. *Annals of Operations Research 16*(1), 201–240. (Zitiert auf den Seiten 6, 33, 39 und 40)

Böttcher, J. (1995). *Projektplanung: Ein exakter Algorithmus zur Lösung des Problems mit partiell erneuerbaren Ressourcen*. Diplomarbeit, Universtität Kiel. (Zitiert auf den Seiten 14, 141, 168 und 169)

Böttcher, J., A. Drexl, R. Kolisch und F. Salewski (1999). Project scheduling under partially renewable resource constraints. *Management Science 45*(4), 543–559. (Zitiert auf den Seiten 14, 17, 18, 100, 122, 127, 130, 141, 168 und 169)

Briskorn, D. und M. Fliedner (2012). Packing chained items in aligned bins with applications to container transshipment and project scheduling. *Mathematical Methods of Operations Research 75*(3), 305–326. (Zitiert auf Seite 13)

Christofides, N., R. Alvarez-Valdes und J. M. Tamarit (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research 29*(3), 262–273. (Zitiert auf Seite 123)

De Reyck, B. und W. Herroelen (1998). A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research 111*(1), 152–174. (Zitiert auf den Seiten 86, 92 und 158)

Dorndorf, U., E. Pesch und T. Phan-Huy (2000a). A branch-and-bound algorithm for the resource-constrained project scheduling problem. *Mathematical Methods of Operations Research 52*(3), 413–439. (Zitiert auf Seite 52)

Dorndorf, U., E. Pesch und T. Phan-Huy (2000b). Constraint propagation techniques for the disjunctive scheduling problem. *Artificial Intelligence 122*(1), 189–240. (Zitiert auf den Seiten 52, 71 und 72)

Dorndorf, U., E. Pesch und T. Phan-Huy (2000c). A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. *Management Science 46*(10), 1365–1384. (Zitiert auf Seite 52)

Drexl, A., J. Juretzka und F. Salewski (1993). Academic course scheduling under workload and changeover constraints. *Manuskripte aus den Instituten für Betriebswirtschaftslehre 337, Universität Kiel*. (Zitiert auf Seite 12)

Drexl, A. und F. Salewski (1997). Distribution requirements and compactness constraints in school timetabling. *European Journal of Operational Research 102*(1), 193–214. (Zitiert auf Seite 12)

Fest, A., R. H. Möhring, F. Stork und M. Uetz (1999). Resource-constrained project scheduling with time windows: A branching scheme based on dynamic release dates. *Technical Report 596, revised version, Technische Universität Berlin.* (Zitiert auf den Seiten 92 und 158)

Franck, B. (1999). *Prioritätsregelverfahren für die ressourcenbeschränkte Projektplanung mit und ohne Kalender.* Aachen: Shaker. (Zitiert auf den Seiten 29, 41 und 42)

Franck, B., K. Neumann und C. Schwindt (2001a). Project scheduling with calendars. *OR Spektrum 23*(3), 325–334. (Zitiert auf den Seiten 30, 32 und 42)

Franck, B., K. Neumann und C. Schwindt (2001b). Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling. *OR Spektrum 23*(3), 297–324. (Zitiert auf den Seiten 73, 99, 115, 116 und 128)

Garey, M. R. und D. S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: Freeman. (Zitiert auf Seite 34)

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics 45*(7), 733–750. (Zitiert auf Seite 13)

Kaplan, L. (1988). *Resource-Constrained Project Scheduling with Preemption of Jobs.* Unveröffentlichte PhD Dissertation, Universität Michigan. (Zitiert auf Seite 125)

Kellner, M., N. Boysen und M. Fliedner (2012). How to park freight trains on rail–rail transshipment yards: The train location problem. *OR Spectrum 34*(3), 535–561. (Zitiert auf Seite 13)

Klein, R. (2000). *Scheduling of Resource-Constrained Projects.* Boston: Kluwer. (Zitiert auf den Seiten 123 und 125)

Klein, R. und A. Scholl (1999). Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. *European Journal of Operational Research 112*(2), 322–346. (Zitiert auf Seite 73)

Knust, S. (2010). Scheduling non-professional table-tennis leagues. *European Journal of Operational Research 200*(2), 358–367. (Zitiert auf Seite 13)

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research 90*(2), 320–333. (Zitiert auf den Seiten 99 und 116)

Kolisch, R. und A. Sprecher (1997). PSPLIB - A project scheduling problem library. *European Journal of Operational Research 96*(1), 205–216. (Zitiert auf Seite 128)

Kolisch, R., A. Sprecher und A. Drexl (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science 41*(10), 1693–1703. (Zitiert auf Seite 129)

Kreter, S. (2016). *Projektplanung mit Kalendern: Struktureigenschaften und Lösungsmethoden*. Aachen: Shaker. (Zitiert auf den Seiten 28, 33 und 51)

Kreter, S., J. Rieck und J. Zimmermann (2016). Models and solution procedures for the resource-constrained project scheduling problem with general temporal constraints and calendars. *European Journal of Operational Research 251*(2), 387–403. (Zitiert auf Seite 51)

Laborie, P. (2003). Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence 143*(2), 151–188. (Zitiert auf Seite 18)

Murty, K. G. (1968). Letter to the editor - An algorithm for ranking all the assignments in order of increasing cost. *Operations Research 16*(3), 682–687. (Zitiert auf Seite 84)

Neumann, K. und M. Morlock (2002). *Operations Research* (2. Auflage). München: Hanser. (Zitiert auf den Seiten 7 und 77)

Neumann, K., H. Nübel und C. Schwindt (2000). Active and stable project scheduling. *Mathematical Methods of Operations Research 52*(3), 441–465. (Zitiert auf Seite 102)

Neumann, K. und C. Schwindt (2003). Project scheduling with inventory constraints. *Mathematical Methods of Operations Research 56*(3), 513–533. (Zitiert auf den Seiten 1, 18, 33 und 34)

Neumann, K., C. Schwindt und N. Trautmann (2005). Scheduling of continuous and discontinuous material flows with intermediate storage restrictions. *European Journal of Operational Research 165*(2), 495–509. (Zitiert auf Seite 18)

Neumann, K., C. Schwindt und J. Zimmermann (2003). *Project Scheduling with Time Windows and Scarce Resources*. Berlin: Springer. (Zitiert auf den Seiten 7, 19, 32, 33, 99, 115 und 158)

Okubo, H., T. Miyamoto, S. Yoshida, K. Mori, S. Kitamura und Y. Izui (2015). Project scheduling under partially renewable resources and resource consumption during setup operations. *Computers & Industrial Engineering 83*, 91–99. (Zitiert auf Seite 13)

Pritsker, A. A. B., L. J. Watters und P. M. Wolfe (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science 16*(1), 93–108. (Zitiert auf Seite 122)

Sampson, S. E. und E. N. Weiss (1993). Local search techniques for the generalized resource constrained project scheduling problem. *Naval Research Logistics 40*(5), 665–675. (Zitiert auf Seite 15)

Sankaran, J. K., D. Bricker und S.-H. Juang (1999). A strong fractional cutting-plane algorithm for resource-constrained project scheduling. *International Journal of Industrial Engineering: Theory, Applications and Practice 6*(2), 99–111. (Zitiert auf Seite 123)

Schirmer, A. (1999). *Project Scheduling with Scarce Resources: Models, Methods, and Applications*. Hamburg: Kovač. (Zitiert auf den Seiten 14, 15, 17, 18, 34, 100, 129, 130 und 131)

Schirmer, A. und A. Drexl (2001). Allocation of partially renewable resources: Concept, capabilities, and applications. *Networks 37*(1), 21–34. (Zitiert auf Seite 15)

Schwindt, C. (1996). Generation of resource-constrained project scheduling problems with minimal and maximal time lags. *Technical Report WIOR-489, Universität Karlsruhe*. (Zitiert auf Seite 128)

Schwindt, C. (1998a). Generation of resource-constrained project scheduling problems subject to temporal constraints. *Technical Report WIOR-543, Universität Karlsruhe*. (Zitiert auf Seite 128)

Schwindt, C. (1998b). *Verfahren zur Lösung des ressourcenbeschränkten Projektdauerminimierungsproblems mit planungsabhängigen Zeitfenstern*. Aachen: Shaker. (Zitiert auf Seite 18)

Schwindt, C. (2005). *Resource Allocation in Project Management*. Berlin: Springer. (Zitiert auf den Seiten 18, 19, 33 und 52)

Talbot, F. B. und J. H. Patterson (1978). An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. *Management Science 24*(11), 1163–1174. (Zitiert auf den Seiten 14 und 141)

Thesen, A. (1977). Measures of the restrictiveness of project networks. *Networks 7*(3), 193–208. (Zitiert auf Seite 128)

Watermeyer, K. und J. Zimmermann (2018). A branch-and-bound procedure for the resource-constrained project scheduling problem with partially renewable resources and time windows. In: M. Caramia, L. Bianco und S. Giordani (Hg.), *Proceedings of the 16th International Conference on Project Management and Scheduling (PMS 2018)*, 259–262. Rome: TexMat. (Zitiert auf den Seiten 2 und 16)

Watermeyer, K. und J. Zimmermann (2020). A branch-and-bound procedure for the resource-constrained project scheduling problem with partially renewable resources and general temporal constraints. *OR Spectrum 42*(2), 427–460. (Zitiert auf den Seiten 3, 16, 36, 78, 85, 123, 128 und 132)

Zimmermann, J. (2001). *Ablauforientiertes Projektmanagement: Modelle, Verfahren und Anwendungen*. Wiesbaden: Gabler. (Zitiert auf Seite 7)

Zimmermann, J., C. Stark und J. Rieck (2010). *Projektplanung: Modelle, Methoden, Management* (2. Auflage). Berlin: Springer. (Zitiert auf den Seiten 7 und 14)

Anhang

A.1 Vergleich der Verfahren zur Einschränkung der Einplanungszeitpunkte für das konstruktionsbasierte Branch-and-Bound-Verfahren

In Tabelle A.1 wird die Performance des konstruktionsbasierten Branch-and-Bound-Verfahrens für das UBOπ-Testset gezeigt, falls entweder Algorithmus 5.7 (CBB1) oder Algorithmus 5.8 (CBB2) zur Bestimmung der eingeschränkten Menge der Einplanungszeitpunkte T_i verwendet wird. Wie den Ergebnissen in Tabelle A.1 zu entnehmen ist, zeigt CBB1 unabhängig von den Instanzgrößen über alle Testsets UBO$n\pi$ eine bessere Performance als CBB2, wobei allerdings nur geringe Abweichungen festgestellt werden können.

		#nTriv	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$
UBO10π	CBB1	693	534	534	159	0	0,067 s	0,056 s
	CBB2		534	534	159	0	0,068 s	0,056 s
UBO20π	CBB1	621	537	581	40	0	7,846 s	0,702 s
	CBB2		535	581	40	0	7,354 s	0,721 s
UBO50π	CBB1	527	183	491	5	31	13,774 s	26,827 s
	CBB2		183	491	5	31	14,082 s	26,911 s
UBO100π	CBB1	484	88	472	0	12	28,693 s	–
	CBB2		88	472	0	12	31,158 s	–
UBO200π	CBB1	466	96	458	0	8	36,908 s	–
	CBB2		95	458	0	8	38,084 s	–

Tabelle A.1: Vergleich der Verfahren zur Einschränkung der Einplanungszeitpunkte für das UBOπ-Testset

A.2 Untersuchung des reimplementierten Branch-and-Bound-Verfahrens

In den Tabellen A.2 und A.3 wird der Einfluss der Zulässigkeitsschranken FB1 und FB2 auf die Performance des Branch-and-Bound-Verfahrens von Böttcher (1995) bzw. Böttcher et al. (1999) untersucht. Das Branch-and-Bound-Verfahren wird für jedes Testset zunächst in einer Basisvariante (BV) ohne Verwendung einer der Zulässigkeitsschranken ausgeführt. Zur Bewertung des Einflusses der Zulässigkeitschranken wird die Basisvariante entweder nur mit FB1, nur mit FB2 oder mit beiden Zulässigkeitschranken (BOT) ausgeführt.

		#nTriv	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$
P10π	BV	2108	827	827	1281	0	0,078 s	0,050 s
	FB1		827	827	1281	0	0,020 s	0,019 s
	FB2		827	827	1281	0	0,081 s	0,053 s
	BOT		827	827	1281	0	0,023 s	0,023 s
P15π	BV	204	134	180	15	9	14,563 s	11,168 s
	FB1		179	181	16	7	3,602 s	0,372 s
	FB2		139	181	15	8	11,644 s	8,078 s
	BOT		181	181	16	7	2,727 s	0,036 s
P20π	BV	165	57	138	10	17	32,188 s	8,436 s
	FB1		134	139	17	9	4,466 s	18,884 s
	FB2		68	137	11	17	24,421 s	7,003 s
	BOT		136	139	16	10	3,974 s	2,187 s
P25π	BV	136	27	110	3	23	7,854 s	5,820 s
	FB1		104	111	10	15	0,478 s	35,119 s
	FB2		41	110	4	22	7,637 s	49,855 s
	BOT		105	111	11	14	0,465 s	25,457 s
P30π	BV	122	16	96	1	25	8,831 s	0,009 s
	FB1		97	104	2	16	2,047 s	0,704 s
	FB2		28	96	2	24	7,823 s	20,477 s
	BOT		98	104	3	15	2,095 s	0,196 s

Tabelle A.2: Performance des Branch-and-Bound-Verfahrens von Böttcher (1995) bzw. Böttcher et al. (1999) in Abhängigkeit der Verwendung der Zulässigkeitsschranken FB1 und FB2 für das Böttcher-Testset

Aus den Ergebnissen der Tabellen A.2 und A.3 für das Böttcher- und das Schirmer-Alvarez-Valdes-Testset geht hervor, dass beide Zulässigkeitsschranken FB1 und FB2 die Performance der Basisvariante mit Ausnahme von Testset P10π durchgängig verbessern können. Für die Testsets mit mehr als zehn realen Vorgängen ergibt sich weiterhin eine Dominanz der Ausführungsvariante FB1 gegenüber FB2, die mit steigender Instanzgröße

		#nTriv	#opt	#zul	#unl	#unb	$\varnothing^{\text{cpu}}_{\text{opt}}$	$\varnothing^{\text{cpu}}_{\text{unl}}$
J10π	BV	808	802	802	5	1	1,315 s	16,261 s
	FB1		802	802	5	1	0,164 s	0,878 s
	FB2		802	802	5	1	0,981 s	1,664 s
	BOT		802	802	5	1	0,171 s	0,041 s
J20π	BV	565	398	552	0	13	8,632 s	–
	FB1		505	560	0	5	6,724 s	–
	FB2		426	555	0	10	4,284 s	–
	BOT		509	561	0	4	6,436 s	–
J30π	BV	453	273	434	0	19	4,405 s	–
	FB1		340	436	0	17	3,897 s	–
	FB2		288	434	0	19	4,214 s	–
	BOT		345	435	0	18	5,573 s	–
J40π	BV	386	181	360	0	26	5,862 s	–
	FB1		260	362	0	24	4,380 s	–
	FB2		192	361	0	25	4,293 s	–
	BOT		261	363	0	23	4,376 s	–
J60π	BV	346	145	306	0	40	3,454 s	–
	FB1		184	309	0	37	1,415 s	–
	FB2		153	308	0	38	4,402 s	–
	BOT		186	309	0	37	2,502 s	–

Tabelle A.3: Performance des Branch-and-Bound-Verfahrens von Böttcher (1995) bzw. Böttcher et al. (1999) in Abhängigkeit der Verwendung der Zulässigkeitsschranken FB1 und FB2 für das Schirmer-Alvarez-Valdes-Testset

zunimmt. Abschließend kann festgestellt werden, dass BOT die besten Ergebnisse über alle Testsets außer für P10π erzielt, wobei zu beachten ist, dass die Ausführungsvariante FB1 für das Testset P20π eine weitere Testinstanz als unlösbar verifizieren kann und für J30π eine zusätzliche Testinstanz als lösbar identifiziert.